Univerza v Ljubljani

Fakulteta za elektrotehniko

Zoran Ivanić

RADIO-FREKVENČNI VIR S SLABILCEM 35 MHz - 4,4 GHz

Diplomsko delo univerzitetnega študija

Mentor: prof. dr. Matjaž Vidmar

Ljubljana, 2016

Zahvala

Zahvaljujem se mentorju prof. Matjažu Vidmarju za podporo in napotke pri izdelavi diplomskega dela.

Zahvaljujem se vsem, ki ste me med celotnim študijskim delom kakorkoli podpirali, navdihovali in me usmerjali na pravo pot.

Posebej se zahvaljujem družini, ki mi je ves čas stala ob strani.

Vsebina

Seznam slik in tabel

Seznam kratic in izrazov

Balun	*Balanced to Unbalanced* – simetrirni transformator
CE	*Chip Enable* – vhod, ki omogoči integrirano vezje
CLK	*Clock* - ura
CMOS	Complementary metal-oxide–semiconductor – komplementarni polprevodnik s kovinskim oksidom
DATA	podatkovni vhod
Fractional PLL	ulomkovna fazno sklenjena zanka
GaAs	galijev arzenid
Integer PLL	celoštevilska fazno sklenjena zanka
LCD	*Liquid crystal display* – zaslon s tekočimi kristali
LE	Load Enable – vhod zapaha, omogoči pošljanje podatkov
LSB	Least Significat Bit – bit z najmanjšo utežjo
MMIC	*Monolithic Microwave Integrated Circuit* – monolitno mikrovalovno integrirano vezje
MSB	Most Significant Bit – bit z največjo utežjo
MUX	multipleksorski izhod
NPN	vrsta tranzistorja
PDRF	*Power Down RF* – vhod, ki izključi visokofrekvenčni izhod
PFD	*Phase Frequency Detector* – fazno-frekvenčni primerjalnik
PLL	*Phase-locked Loop* – fazno sklenjena zanka
RF+, RF-	diferencialna visokofrekvenčna izhoda
SPI	*Serial Peripheral Interface* – serijski periferni vmesnik
špica	*Spur* – moteča frekvenčna komponenta v spektru
TCXO	*Temperature Compensated Crystal Oscillator* – temperaturno kompenzirani kristalni oscilator
VCO	*Voltage Controlled Oscillator* – napetostno kontroliran oscilator
VF	visokofrekvenčni
via	Povezava med zgornjo in spodnjo stranjo vezja

Seznam simbolov

veličina/oznaka		**enota**	
ime	**simbol**	**ime**	**simbol**
frekvenca	f	hertz	Hz ali s^{-1}
čas	t	sekunda	s
jakost signala, moč	P	decibel-miliwatt ali decibel-nosilec ali decibel	dBm ali dBc ali dB
jakost faznega šuma	P	decibel-nosilec/hertz	dBc/Hz
napetost	U	volt	V
upornost	R	ohm	Ω
kapacitivnost	C	farad	F
induktivnost	L	henry	H

Povzetek

To delo opisuje izdelavo radio-frekvenčnega vira, ki deluje na osnovi ulomkovne fazno-sklenjene zanke za območje od 34,375 MHz do 4400 MHz. Z dodanim nastavljivim slabilcem dobi tak vir obliko praktičnega instrumenta, ki se uporablja pri razvoju in testiranju radijskih komponent in zvez.

V uvodnem delu je opisana zasnova instrumenta z izbranimi komponentami. Sledi kratek teoretični opis sinteze signala s formulami z uporabo ulomkovne fazno-sklenjene zanke na primeru ADF4351 integriranega vezja.

V drugem delu je opisana praktična izvedba vseh sestavnih blokov inštrumenta z električnimi načrti, meritvami karakteristik in se zaključi z opisom pomembnejših delov programske kode, ki krmili instrument. ADF4351 se napaja preko ADP150 in LM1117 regulatorjev. Sintetizirani signal potuje preko baluna iz feritnih obročkov na MMIC ojačevalnik gali5+ in od tu naprej na vezje slabilca. Slabilec je zgrajen z zaporedno vezavo dveh HMC307, ki se krmilita z dvemi preklopniki 4053 in serijsko-paralelnega pomikalnega registra 74HC595. HMC307 in 4053 potrebujejo za svoje delovanje negativno napetost, ki jo zagotavlja inverter LM828. Pomikalni register se krmili z mikrokrmilnikom, ki sprejema ukaze preko štirih tipk s pomočjo LCD zaslona. Instrument se napaja preko regulatorja RC1587M, ki pretvarja zunanjo napetost +12 V na +5 V. Programska koda omogoča preprosto uporabniško izkušnjo, natančno kalibracijo instrumenta in preverjanje delovanja kode preko serijskega vodila.

V zadnjem delu so opisane še nekatere omejitve ADF4351 in ideje za izboljšave, ter predstavljeni rezultati meritev moči izhodnega nosilca.

Ključne besede: ADF4351, HMC307, 4053, 74HC595, LM828, ADP150, LM1117, fazno-sklenjena zanka, napetostno kontrolirani oscilator, slabilec

Abstract

Radio-frequency source with an attenuator 35MHz – 4.4GHz

This work describes the design of a radio-frequency source, based on fractional phase-locked loop signal synthesis in the range of 34.375MHz to 4400MHz. With the addition of an adjustable attenuator to the RF source, we obtain a practical instrument that can be used in the development and testing of radio components and networks.

The introductory part describes the design of the instrument with the selected components. Followed by a brief description of the theoretical signal synthesis formulas, using fractional phase-locked loop in the case of the ADF4351 integrated circuit.

The second part describes the practical implementation of all the constituent blocks of the instrument with the electrical plans, measurement characteristics and concludes with a description of the most important parts of the software code that control the instrument. ADF4351 is powered by ADP150 and LM1117 regulators. The synthesized signal travels through the balun made of ferrite rings, to the MMIC amplifier Gali5 + and from there, to the circuit of the attenuator. The attenuator is made of two HMC307s, connected in series, which are controlled by two 4053 multiplexers and a serial-parallel shift register 74HC595. The HMC307 and 4053 need a negative voltage supply, for their operation, which is provided by the LM828 inverter. The shift register is controlled by a microcontroller, which receives commands via the four keys on the LCD screen. The instrument is powered by the voltage regulator RC1587M that converts an external voltage of + 12V to + 5V. The software code provides a better user experience, accurate instrument calibration and verification of operation code via the serial bus.

The final section describes some limitations of the ADF4351 and ideas for improvement, and the results of measurements of the output power of the carrier.

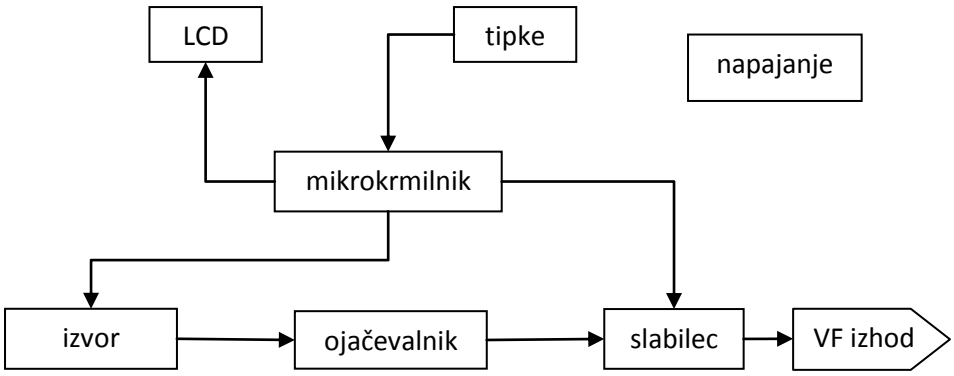
Keywords: ADF4351, HMC307, 4053, 74HC595, LM828, ADP150, LM1117, phase-locked loop, voltage controlled oscillator, attenuator, VCO, PLL

Uvod

Radio-frekvenčni vir signala je nepogrešljiv instrument v vsakem laboratoriju, ki se ukvarja z visokofrekvenčno (VF) tehniko. Pomaga pri razvoju in testiranju radijskih komponent, kot so ojačevalniki, sita, modulatorji, testiranju sprejemnikov, radijskih zvez, itd. Signal se lahko generira na več načinov. Eden takih načinov je sinteza stabilnega frekvenčnega signala s pomočjo napetostno kontroliranega oscilatorja (angl. Voltage Controlled Oscillator – VCO) in fazno-sklenjene zanke (angl. Phase-locked Loop – PLL), ki se ponavadi uporablja v komunikacijski napravah. Generirani signal se nato vodi na vhod neke druge testne naprave in mora biti po jakosti prilagojen tej napravi, da ne pride do poškodbe notranjih komponent naprave. Nivo vhodnega signala se prilagodi s slabilcem, ki se ga veže med vir signala in vhod testne naprave. Slabilec je naprava, ki oslabi moč signala pri čemer se oblika signala bistveno ne spremeni.

Zasnova instrumenta

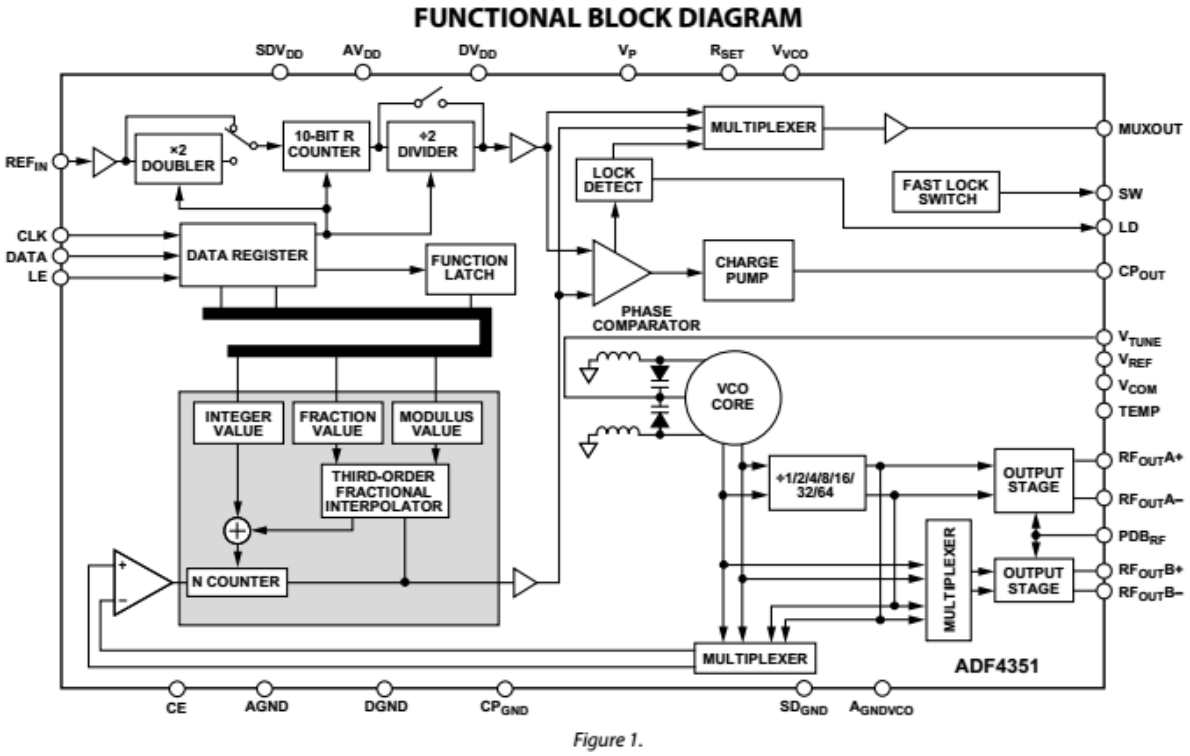
Za diplomsko delo sem dobil precej praktično nalogo. Raziskati je bilo potrebno možnosti za izdelavo radio-frekvenčnega vira za uporabo pri laboratorijskem delu in instrument izdelati. Instrument je zasnovan okoli mikrokrmilnika, ki digitalno nastavlja vir signala, kateremu je zaporedno vezan digialno nastavljiv slabilec signala. Kot jedro vira signala je bil izbran širokopasovni frekvenčni sitentizator starejše generacije ADF4351 proizvajalca Analog Devices z vgrajenima PLL in VCO. Za slabljenje generiranega signala sem dobil na voljo par starejših HMC307 slabilcev s slabljenjem od 0dB do 31 dB po korakih 1 dB. Glede na to, da ima ADF4351 diferencialni izhod je smiselno oba izhoda povezati preko baluna v enojni izhod. Balun sem naredil iz poltrdega koaksialnega kabla UT-047, izhodni signal pa nato še ojačal s širokopasovnim MMIC ojačevalnikom Gali5+. Ojačeni signal nato potuje na vhod dveh zaporedno vezanih slabilcev HMC307, kjer se lahko oslabi od 0 do 62 dB po 1 dB korakih. Vse skupaj se krmili s pomočjo mikrokrmilnika Arduino Mega 2650 preko štirih tipk in 16x2 vrstičnega LCD zaslona HD44780, ki izpisuje nastavljeno frekvenco in moč generiranega nosilca v dBm.



Slika 1: Blokovni načrt instrumenta

Osnove sinteze signala

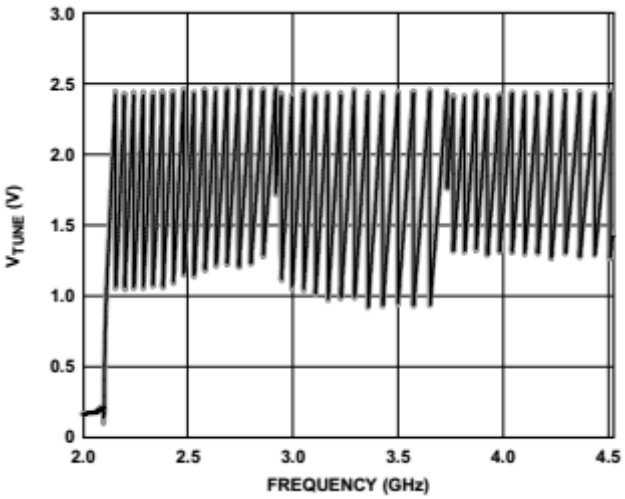
ADF4351 vsebuje sintetizator signala, ki deluje na osnovi PLL in VCO. Za sintezo signala potrebujemo referenčni vir, ki je ponavadi generiran s pomočjo TCXO, ki zagotavlja temperaturno stabilnost reference. Referenčni signal potuje na števec R, ki lahko poskrbi, da primerjalno frekvenco delimo z vrednostjo R. V primeru tega instrumenta je R nastavljen na 1 in primerjalna frekvenca je enaka referenčni. Poleg R števca se nahaja še vezje, ki lahko referenčno frekvenco podvoji ali prepolovi, ki pa v tem primeru ni v uporabi. Na spodnji sliki je prikazan blokovni načrt integriranega vezja.



Slika 2: Funkcionalni diagram ADF4351

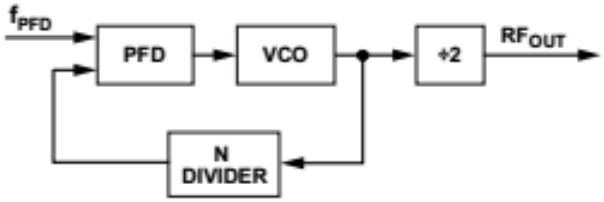
Napetostno kontroliran oscilator

VCO, eden od gradnikov znotraj PLL zanke je sestavljen iz treh posameznih VCO, kjer vsak uporablja 16 območij, ki se med seboj prekrivajo tako, da skupaj pokrivajo frekvenčno območje od 2200 MHz do 4400 MHz.



Slika 3: Graf VCO

Za delovanje instrumenta sem izbral način, ko je povratna zanka v osnovnem (angl. fundamental) načinu, kar pomeni, da je zanka sklenjena pri izhodu iz VCO, pred izhodnim delilnikom, kot je to prikazano na sliki 4.



Slika 4: Osnovni način PLL zanke

N delilnik

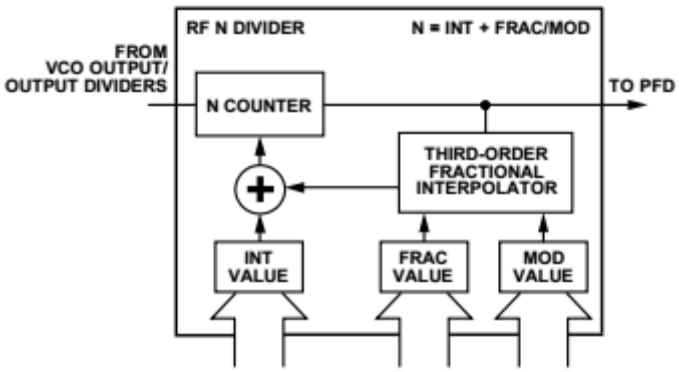
Naslednji pomembni gradnik znotraj PLL je N delilnik (slika 5), s katerim s pomočjo vpisa v registre INT, FRAC in MOD nastavimo izhodno frekvenco VCO po enačbi

$$f_{vco} = f_{PFD} \times (INT + \frac{FRAC}{MOD}) \quad (1)$$

kjer je f_{PDF} primerjalna frekvenca. N delilnik vsebuje ulomkovni interpolator (Σ-Δ modulator), ki je krivec za nezaželjene špice v spektru generiranega signala, ko integrirano vezje deluje v ulomkovnem načinu. Špice se v najslabšem primeru ponavljajo na interval:

$$\frac{f_{PFD}}{L} \quad (2)$$

kjer je v števcu primerjalna frekvenca v imenovalcu pa dolžina ponavljajočega kodnega zaporedja v Σ-Δ modulatorju, katerega lahko izračunamo s pomočjo tabele št. 7 v podatkovnem listu [1] in je odvisna od velikosti imenovalca MOD.



Slika 5: N-delilnik

Primerjalna frekvenca in velikost MOD registra določata ločljivost oz. frekvenčni korak, ki ga izračunamo po formuli:

$$r_{kanala} = \frac{f_{PFD}}{MOD} \quad (3)$$

Vrsta N delilnika določa tudi tip PLL. V kolikor je N delilnik celoštevilski, govorimo o celoštevilski PLL (angl. Integer PLL), pri katerih je najmanjši možni frekvenčni korak enak velikosti primerjalne frekvence.

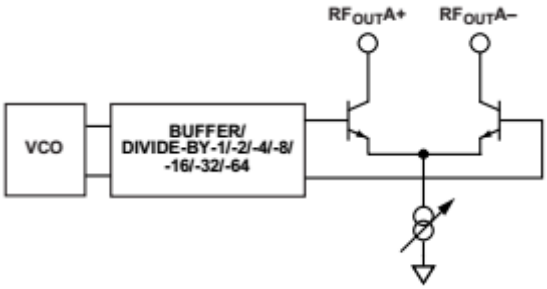
Če je MOD = 4000 in $f_{PFD} = 32\ MHz$ imamo v najslabšem možnem primeru (pri najvišjih generiranih frekvencah) najmanjši možen korak 8 kHz pri 12-bitnem MOD registru, kjer se vrednosti

MOD registra raztezajo med 2 in 4095. Za uporabo v generatorju sem se odločil za izbiro najvišje možne primerjalne frekvence. Prispevek h faznem šumu zaradi primerjalne frekvence narašča z $10 \times \log(f_{PFD})$. Prispevek h faznemu šumu zaradi velikosti delilnika N narašča z $20 \times \log(N)$ [21]. Z vsakim podvojevanjem primerjalne frekvence imamo lahko polovico manjši N delilnik, kar pomeni, da prispevek šuma zaradi N delilnika pade za 3 dB. Tretja prednost najvišje možne primerjalne frekvence je tudi to, da so špice, ki jih povzroča referenčni vir bolj oddaljene od nosilnega signala.

Izhodna stopnja

Z uporabo izhodnega delilnika, ki deli v razmerju D = { 1, 2, 4, 8, 16, 32, 64 } lahko osnovno frekvenco VCO delimo navzdol do najnižje izhodne 34,375 MHz.

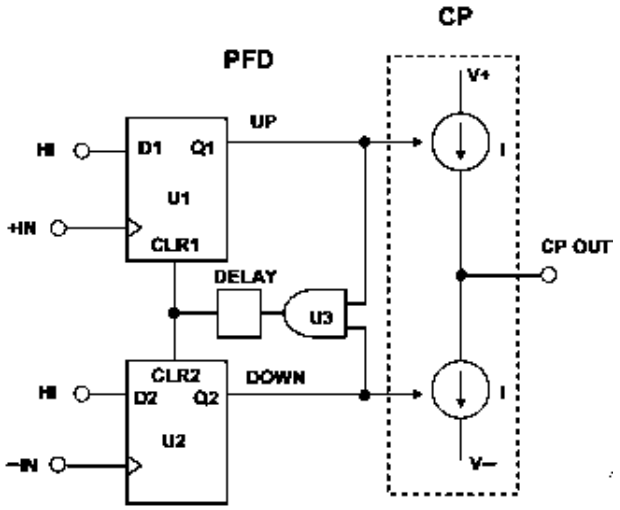
$$f_{IZHOD} = \frac{f_{VCO}}{D} \tag{4}$$



Slika 6: Izhodna stopnja

Črpalka nabojev in fazno frekvenčni primerjalnik

Izhod iz N delilnika in izhod iz referenčnega vira sta povezana na vhod fazno-frekvenčnega primerjalnika PFD, katerega izhod je proporcionalen frekvenčni in fazni razliki med njima. Izhodni del PFD je sestavljen iz tokovne črpalke nabojev, ki pretvarja logična stanja PFD v analogen signal, ki je primeren za krmiljenje VCO preko zunanjega zančnega sita.



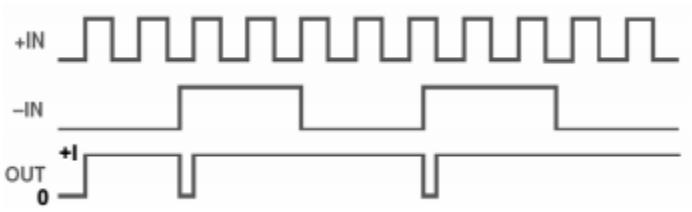
Slika 7: Fazno-frekvnčni primerjalnik

Poenostavljena PFD implementacija z dvemi flip-flopi D in zakasnilnim vezjem (DELAY) je prikazana na sliki 7. En Q izhod flip-flopa vklaplja pozitivni tokovni vir, drugi pa negativnega. Izhod črpalke nabojev je glede na stanje obeh flip-flop vhodov predstavljen v spodnji pravilnostni tabeli.

UP	DOWN	CP OUT
1	0	+I
0	1	–I
0	0	0

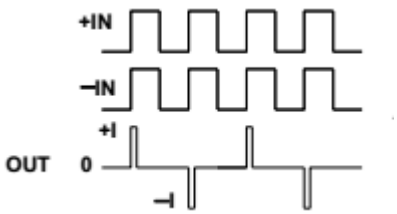
Slika 8: Pravilnostna tabela črpalke

V splošnem primeru, ko PLL zanka ni zaklenjena imamo na vhodih in izhodih fazno-frekvenčnega primerjalni graf podoben temu, na sliki 9.



Slika 9: Splošen graf, ko ni zaklepa

Zakasnilni element v vezju poskrbi, da PFD deluje stabilno. Pri frekvenčno in fazno poravnanih obeh vhodih, dobimo graf na sliki 10.



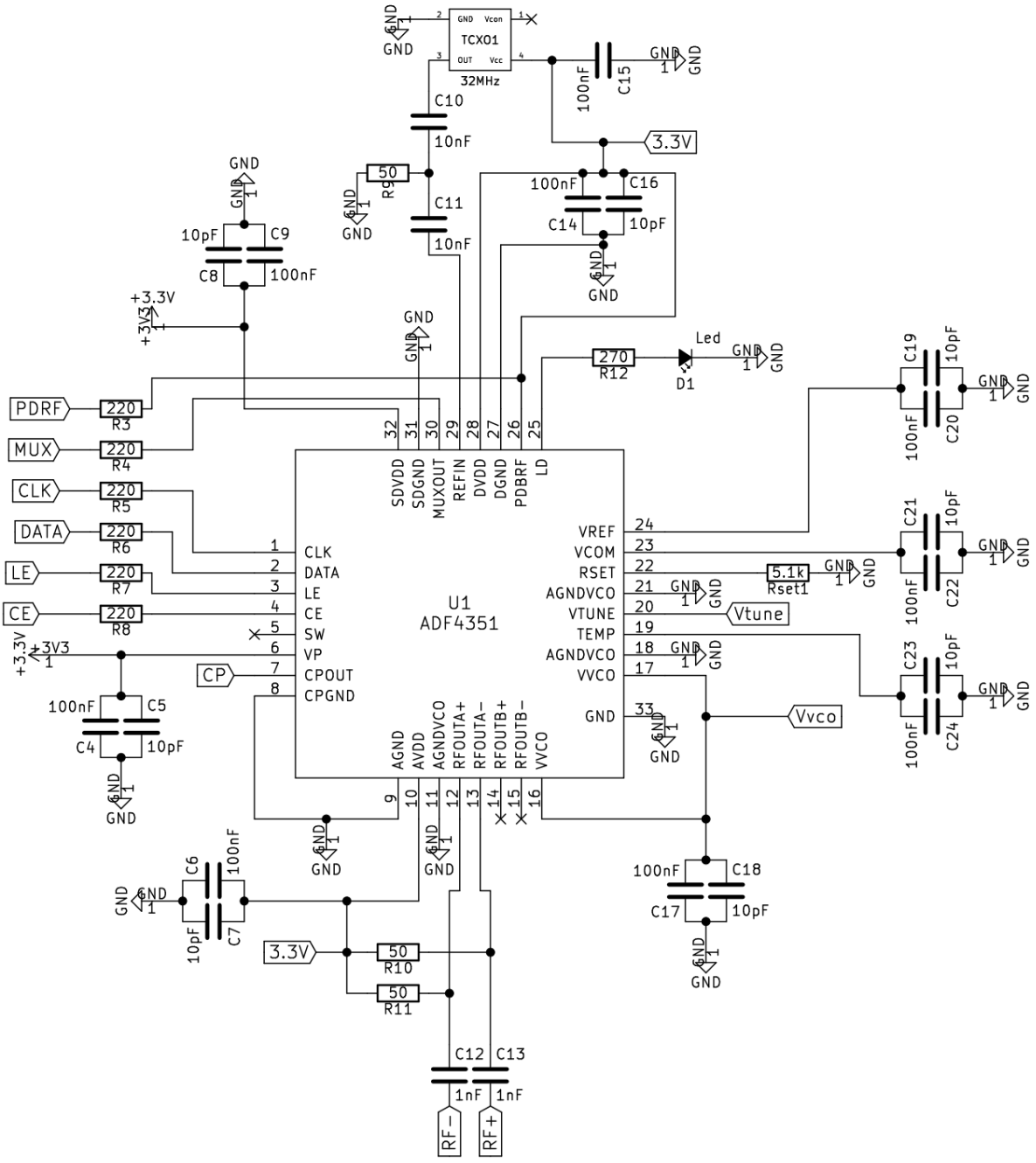
Slika 10: Zaklenjena zanka

Takrat govorimo, da je zanka fazno (in frekvenčno) zaklenjena. Kljub temu, se na izhodu pojavijo pozitivni in negativni impulzi, katerih trajanje je enako zakasnitvi zakasnilnega člena. Trajanje te zakasnitve (angl. antibackslash pulse) je potrebno nastaviti v registru integriranega vezja, glede na to ali deluje v ulomkovnemem ali pa v celoštevilskem načinu. Integrirano vezje ADF4351 vsebuje šest 32-bitnih registrov, katere se programira preko tripolnega serijskega vmesnika.

Opis gradnje

ADF4351 se nahaja v majhnem 5 mm x 5 mm velikem ohišju CP-32-2. Na spodnji strani ima kontakt večje površine, ki mora biti povezan na maso. Napaja s 3,3 V napetostjo in omogoča generiranje izhodne frekvence v obsegu med 34,375 MHz in 4400 MHz.

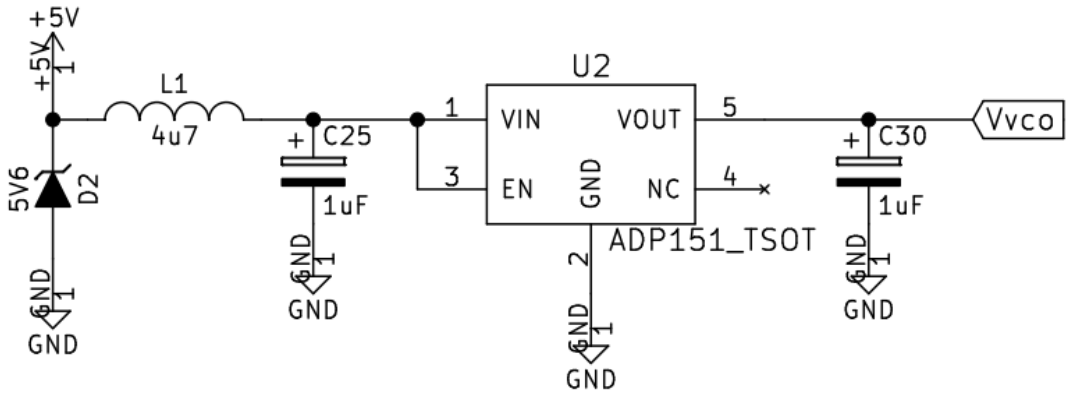
Na referenčni vhod PLL v ADF4351 je iz TCXO preko 50 Ω prilagodilnega upora in dveh 10 nF kondenzatorjev pripeljan signal s frekvenco 32 MHz, ki je v tem primeru tudi enaka primerjalni frekvenci fazno-frekvenčnega detektorja. Okrog integriranega vezja so uporabljeni pari 100 nF in 10 pF blokirnih kondenzatorjev. Po navodilih iz podatkovnega lista so postavljeni čim bližje med izhodnimi priključki in maso z namenom dušenja VF energije na napajalnih linijah. Podobno funkcijo opravljajo tudi 220Ω upori na podatkovnih linijah CLK, DATA, LE, MUX, CE in PDRF, ki so vezani na mikrokrmilnik. Njihov namen je dušenje zvonenja priključnih vodov in omejevanje sevanja radijskih motenj, obenem pa omejujejo morebiten prevelik tok.



Slika 11: Električni načrt vira signala

Napajalnik za VCO

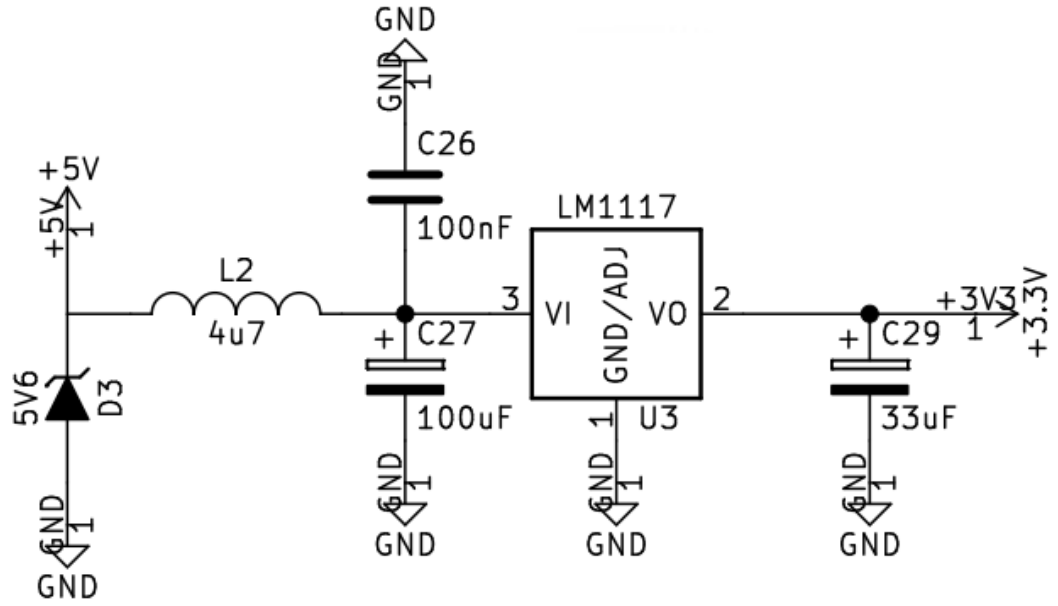
Generator uporablja dva ločena napajalnika. Eden je namenjen napajanju VCO, drugi pa za vse ostale napajalne napetosti ADF4351. Napajanje VCO mora bit čim bolj stabilno in brez šuma, da si zagotovimo čist spekter in čim manjši fazni šum ob nosilcu. Za napajanje na VCO nizko-šumni linearni CMOS regulator ADP150 pretvori +5 V napetost na +3,3 V (slika 12). Za svoje stabilno delovanje potrebuje na vhodu in izhodu sklopni kondenzator vrednosti 1 µF [6]. Vhod je dodatno filtriran s 4,7 µH dušilko in zaščiten s 5,6 V zener diodo.



Slika 12: Napajalnik za VCO

Drugi napajalnik

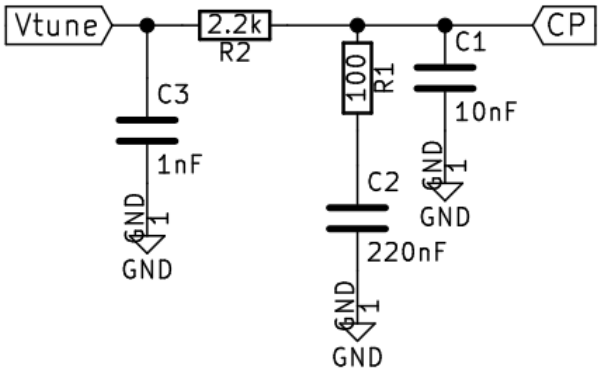
Drugi napajalnik na spodnji sliki je zgrajen okoli linearnega regulatorja LM1117. Vhod je prav tako zaščiten s 5,6 V zener diodo in filtriran s 4,7 µH dušilko. Za glajenje napetosti skrbita tantalova kondenzatorja vrednosti 33 µF in 100 µF.



Slika 13: Drugi napajalnik

Zančno sito

Upor R_{set} na priključku št. 22 določa tok črpalke nabojev I_{cp} po formuli $R_{set} = \frac{25,5}{I_{cp}}$. Z uporom 5,1 Ω nastavimo maksimalni tok na 5 mA. V registru R2 lahko ta tok še dodatno nastavljamo s štirimi biti (šestnajst različnih vrednosti). Povezava iz priključka črpalke naboja je speljana preko zančnega sita PLL na kontrolni vhod priključka VCO. Napetost na tem vhodu določa izhodno frekvenco VCO. Zančno sito tretjega reda sem načrtoval po priporočilih proizvajalca z njihovim orodjem za simulacijo ADIsimPLL [8]. Sito načrtujemo za izbrano primerjalno frekvenco faznega detektorja, ki je v tem primeru 32 MHz. Elemente filtra je smiselno načrtovati za srednjo vrednost možnega toka I_{cp}, torej za 2,5 mA, tok pa nato po potrebi malce povečati ali zmanjšati. Pri tem se je potrebno zavedati, da s spreminjanjem I_{cp} spreminjamo tudi pasovno širino sita.



Slika 14: Zančno sito

Simulacijski program mi je predlagal sito tretjega reda pasovne širine 15 kHz. Po poskušanju različnih dobavljivih vrednosti elementov filtra je nastalo sito s pasovno širino B = 16 kHz in fazno varnostjo ϕ = 47 °.

Loop Filter	CPP_3C
Specify:	Components
Loop Bandwidth	16.0kHz
Phase Margin	47.3 deg
Zero Loc.	7.23kHz
Pole Loc.	67.9kHz
Last Pole	178kHz
C1	10.0nF
R1	100
C2	220nF
R2	2.20k
C3	1.00nF

Slika 15: Simulacija sita pri 16kHz

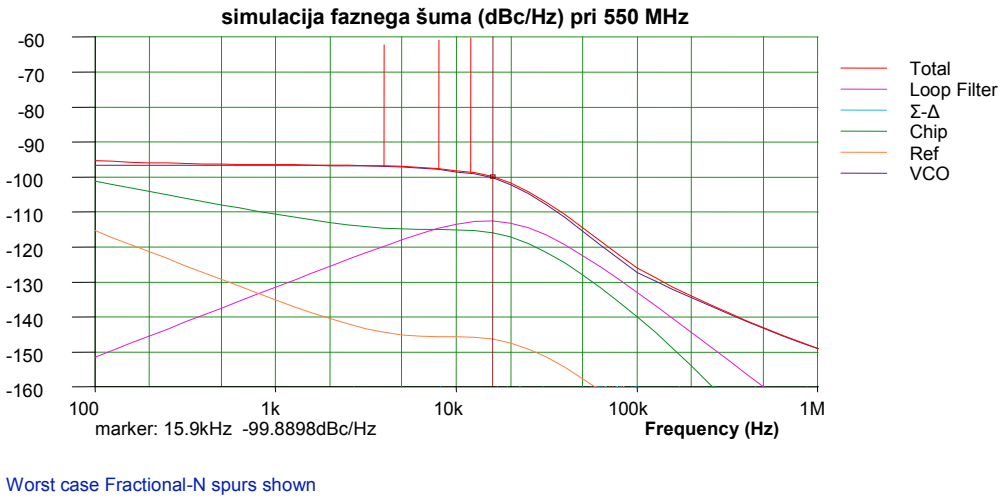
V kolikor sedaj spremenim I_{cp} = 3,44 mA pri R_{set} = 5,1 kΩ (v registru R2 popravim nastavitve ustreznih bitov za črpalko nabojev na B12 = 1 B11 = 0 B10 = 1 B9 = 0) bi moral dobiti naslednje lastnosti filtra:

Loop Filter	CPP_3C
Specify:	Components
Loop Bandwidth	20.8kHz
Phase Margin	47.1 deg
Zero Loc.	7.23kHz
Pole Loc.	67.9kHz
Last Pole	178kHz
C1	10.0nF
R1	100
C2	220nF
R2	2.20k
C3	1.00nF

Slika 16: Simulacija sita pri 21 kHz

Pasovna širina se poveča na 21 kHz.

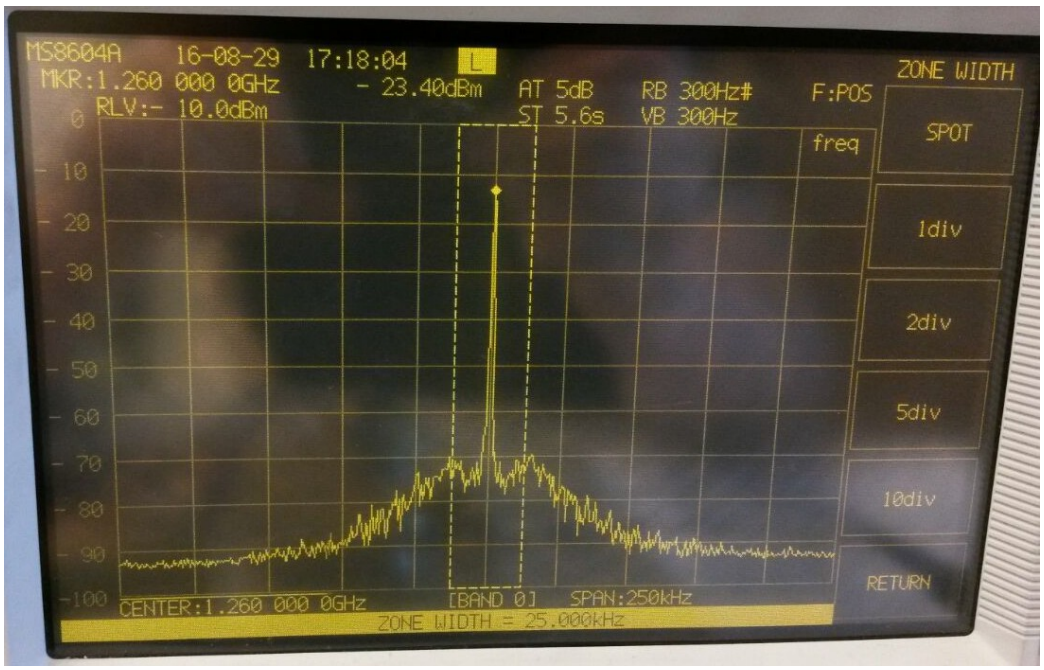
Program za simulacijo omogoča tudi predvidevanje prispevkov šuma različnih virov (integriranega vezja, zančnega sita, Σ-Δ modulatorja, referenčnega vira in VCO), ki se nato prištevajo celotnemu faznemu šumu. V kolikor pri simulaciji izberemo takšno frekvenco, da sintetizator deluje v ulomkovnem načinu, program predvidi tudi najslabši možen primer za prve tri moteče špice, ki so posledica kvantizacijskega šuma Σ-Δ modulatorja. V spodnji simulaciji, okrog 550 MHz, jih je v najslabšem primeru za pričakovati pri 4 kHz, 8 kHz in 12 kHz.



Slika 17: Simulacija faznega šuma

V kolikor je zančno sito pravilno izdelano, da je možno zaklepanje PLL zanke, se mora ob zaklepu na izhodu priključka št. 25 preko 270 Ω upora prižgati LED dioda. LED dioda služi kot indikator zaklepa zanke PLL. Upor 270 Ω omejuje tok diode. Pri testiranju vezja sem preizkušal več sit za različne primerjalne frekvence (10 MHz, 20 MHz, 32 MHz) in pri vseh sem uspel doseči uspešno zaklepanje PLL na celotnem frekvenčnem področju.

Meritev širine sita pri nastavljenem I_{cp} = 1,88 mA pri R_{set} = 5,1 kΩ, ki z zgoraj izbranimi elementi sita simulira pasovno širino B = 12,8 kHz, ϕ = 45,7° kaže, da se simulacija dobro ujema z resničnim sitom. Širina črtkastega pravokotnika na zaslonu je 25 kHz pri frekvenci nosilca f=1260 MHz.



Slika 18: Meritev širine sita pri B=12,8 kHz

Visokofrekvenčna simetrična izhoda sintetizatorja RF+ in RF- sta vezana na kolektorja diferencialnega para NPN tranzistorjev [slika 6], zato izhoda na priključkih prilagodimo z dvema 50 Ω uporoma na +3,3 V. Izhodni signal vodimo na balun preko kondenzatorjev 1 nF, ki so izbrani tako, da ne blokirajo generiranega signala na najnižjem frekvenčnem območju.

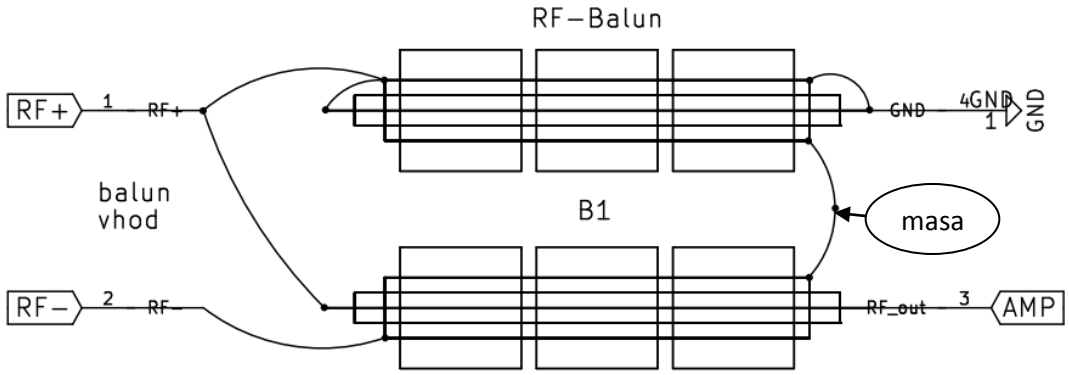
Balun

Balun je izdelan iz valjastih feritnih obročkov tipa FB-43-101 z relativno permeabilnostjo $\mu_r = 800$ in tabelirano impedanco [9]:

Z (Ω)	f (MHz)
17	10
26	25
40	100
56	250

Tabela 1: Karakteristika FB-43-101

Trije feritni obočki so nanizani na vsako stran poltrdega kabla UT-047. Kabli med sabo povezani, kot je to narisano na spodnji sliki.



Slika 19: Načrt baluna

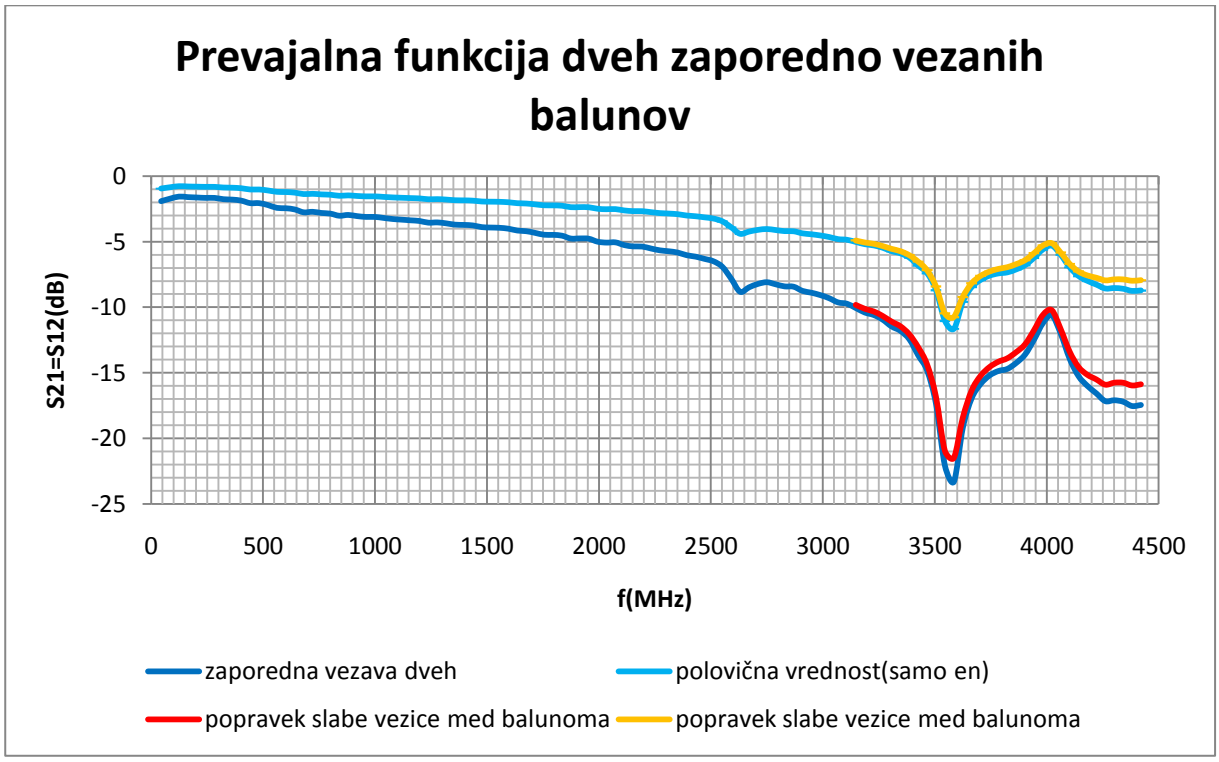
Na izhodu je povezava med obema stranema baluna pricinjena na maso vezja. Namen baluna je, da iz dveh izhodov dobimo enega, ki je tudi čim bolj impedančno prilagojen, feritni obročki pa blokirajo visokofrekvenčne tokove na površini kabla.

Meritev karakteristike baluna

Za meritev karakteristike baluna, je bilo potrebno izdelati dva enaka in jih vezati zaporedno tako, da vežemo simetrični sponki skupaj in imamo prehod iz asimetričnega vhoda na asimetrični izhod.

Slika 20: Testni balun

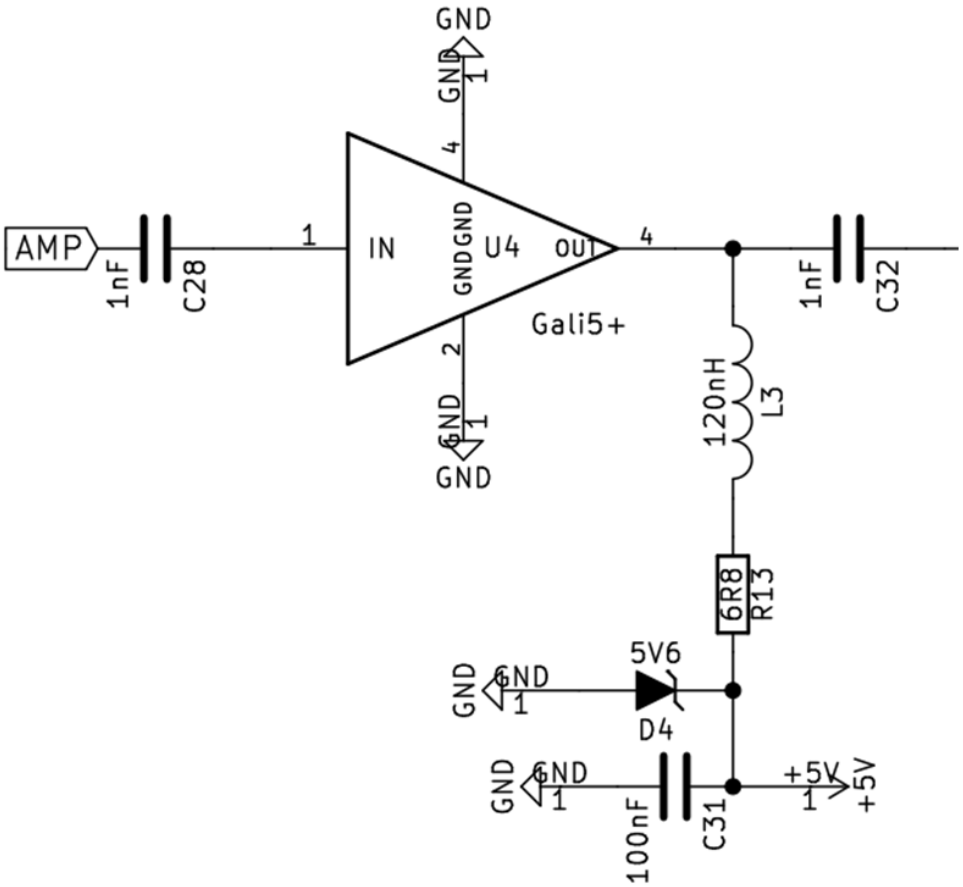
Meritev je bila izvedena na vektorskem analizatorju vezij. Graf kaže, da je balun praktično lepa padajoča premica. Za slabljenje, ki je vidno iz grafa okoli 3500 MHz se je izkazalo, da je posledica vezave balunov med seboj in velike razdalje med veznimi žicami. Dodatne via povezave vzdolž vezave balunov so slabljenje na tem mestu izboljšale še za dobrih 5 dB.



Slika 21: Prevajalna funkcija baluna

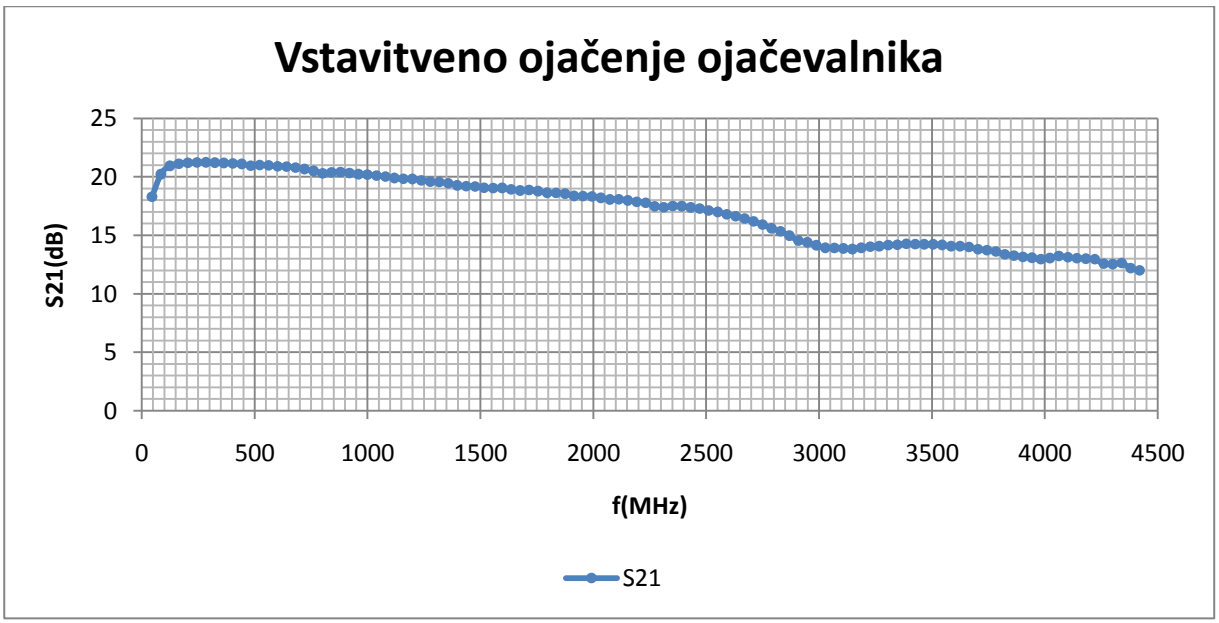
Ojačevalnik

Signal iz baluna potuje naprej preko blokirnega kondenzatorja 1nF na MMIC ojačevalnik Gali5+.



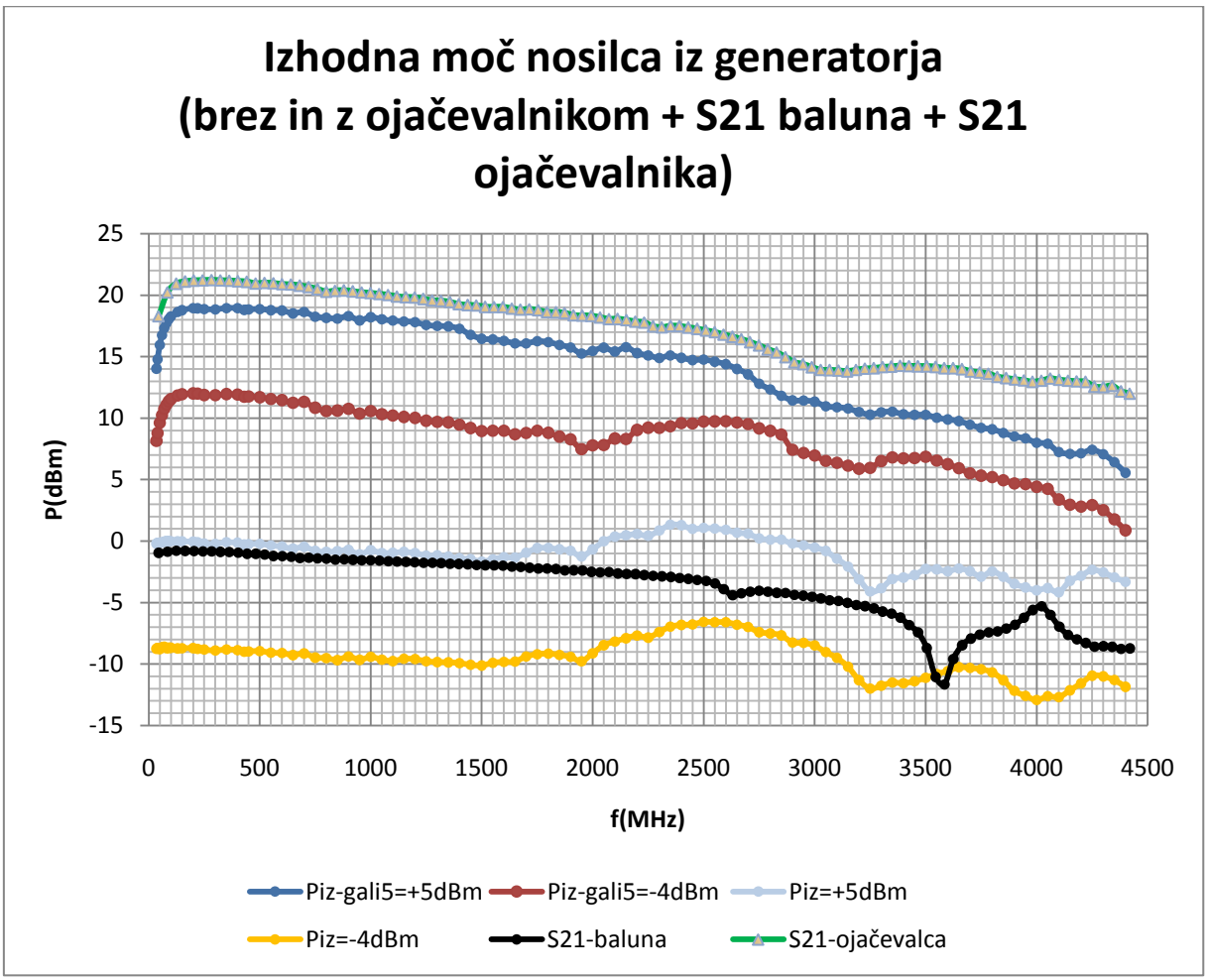
Slika 22: Načrt ojačevalnika

Ojačevalnik je narejen z darlington vezavo tranzistorjev in se nahaja v ohišju SOT-89. Napajalni del vsebuje sklopni kondenzator 100 nF in zaščitno 5,6 V zener diodo. Visokofrekvenčna dušilka služi blokiranju motenj, upor R13 pa določa delovno točko ojačevalnika. Na izhodu je tako kot na vhodu blokirni kondenzator vrednosti 1nF. Ker za napajanje po podatkovnem listu ni predvidena napetost +5 V sem s poskusnimi meritvami delovnega toka in P1dB točke določil vrednost upora na 6,8 Ω. Pri tej vrednosti, odvisno od frekvence, delovni tok dosega največ 80 mA, kar je dovolj daleč od maksimalnega dovoljenega, ki znaša 85 mA. Vstavitveno ojačenje sem izmeril na vektorskem analizatorju vezij pri vhodni moči -20 dBm.



Slika 23: Vstavitveno ojačenje ojačevalnika

Integrirano vezje ADF4351 ima možnost nastavljanja izhodne moči med -4 dBm in +5 dBm. Spodnji graf prikazuje izhodno moč generiranega nosilnega signala preko baluna brez ojačevalnika (rumena in svetlo modra črta) in z ojačevalnikom (modra in rdeča črta). Črna črta je prevajalna funkcija baluna, zgornja pa prevajalna funkcija ojačevalnika. Moč nosilca pri različnih frekvencah in izhodnih močeh je bila merjena na laboratorijskem spektralnem analizatorju.



Slika 24: Primerjalni graf

Laminat

Generator in ojačevalnik sta zgrajena na isti ploščici na laminatu tipa FR4 debeline 0,8 mm, dvostransko, z maso spodaj. Izhodna mikrotrakasta linija impedance 50 Ω iz ojačevalnika je izračunana s pomočjo programa za mikrotrakaste vode v programu KiCad in je široka približno 1,48 mm (58 mil). Laminat FR4 je poceni material, ki slovi po izgubah pri visokih frekvencah. Proizvajalci ponavadi navajajo frekvenčno karakteristiko FR4 samo do 1 GHz. V mojem primeru je bil zanalašč izbran ravno zaradi te lastnosti, z namenom dušenja morebitnih motenj, koristnega signala, pa ne more preveč oslabiti na tako kratkih linijah.

Slika 25: Fotografija visokofrekvenčnega dela signalnega vira

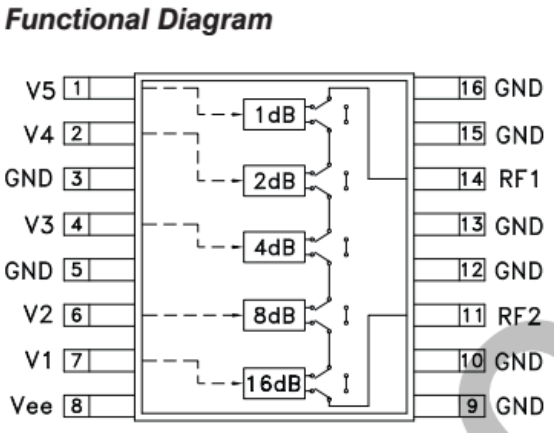
Slabilec

Signal iz ojačevalnika potuje naprej do slabilca. Jedro slabilca predstavljata dva zaporedno vezana HMC307. To je slabilec narejen iz GaAs, ki se nahaja v QSOP ohišju s šestnajstimi stranskimi priključki in dodatnim priključkom za maso na spodnji strani ohišja. Znotraj integriranega vezja se nahaja 5 slabilcev (1 dB, 2 dB, 4 dB, 8 dB in 16 dB) s katerimi lahko nastavljamo vse možne kombinacije po 1 dB med 0 dB in 31 dB. Nastavljenemu slabljenju se prišteva še lastno vstavitveno slabljenje slabilca, ki znaša okoli 2 dB.

Truth Table

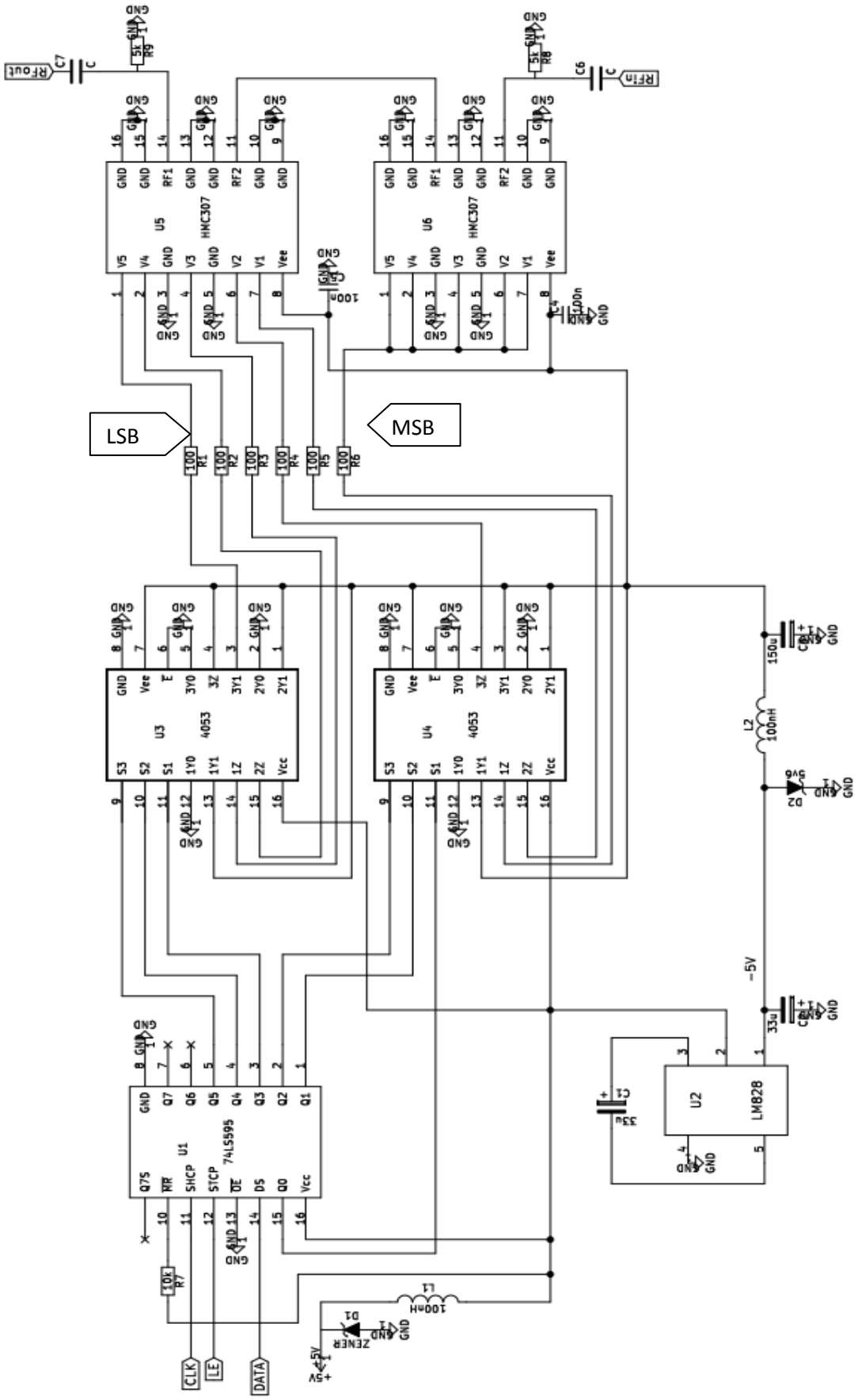
Control Voltage Input					Attenuation State RF1 - RF2
V1 16 dB	V2 8 dB	V3 4 dB	V4 2 dB	V5 1 dB	
Low	Low	Low	Low	Low	Reference I.L.
Low	Low	Low	Low	High	1 dB
Low	Low	Low	High	Low	2 dB
Low	Low	High	Low	Low	4 dB
Low	High	Low	Low	Low	8 dB
High	Low	Low	Low	Low	16 dB
High	High	High	High	High	31 dB Max. Atten.
Any combination of the above states will provide an attenuation approximately equal to the sum of the bits selected.					

Slika 26: Pravilnostna tabela HMC307



Slika 27: Blokovni načrt slabilcev znotraj integriranega vezja

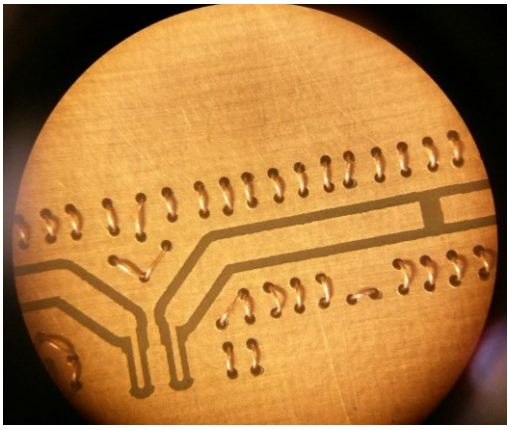
Integrirano vezje HMC307 se krmili in napaja z negativno napetostjo -5 V. Možno ga povezati tudi na pozitivno delovno napetost in frekvenčno karakteristiko uravnavati s sklopnimi kondenzatorji, a je karakteristika primerna za nižje frekvenčno območje do 2 GHz [10]. Za negativno napajanje sem izbral nizkošumni napetostni inverter na osnovi črpalke nabojev LM828. Ta za svoje delovanje potrebuje le dva kvalitetna tantalova kondenzatorja. Negativna napetost se filtrira preko 100 µH dušilke. Tik ob vsakem napajalnem kontaktu obeh slabilcev sem dodal še blokirni 100 nF kondenzator. Poleg slabilcev potrebujejo negativno napajalno napetost tudi preklopniki 4053, katerih namen je preklapljanje stikal v slabilcu. 0 V predstavlja logično ničlo in -5 V predstavlja logično enico Vsako integrirano vezje 4053 vsebuje tri dvokanalne preklopnike. Z dvema integriranima vezjema tako preklapljamo vseh šest stikalnih linij. S prvimi petimi stikalnimi linijami na HMC307 z oznako U5 nastavljamo slabljenje od 0 dB do 31 dB, s šesto linijo na HMC307 z oznako U6 pa razširimo območje slabljenja na 62 dB.



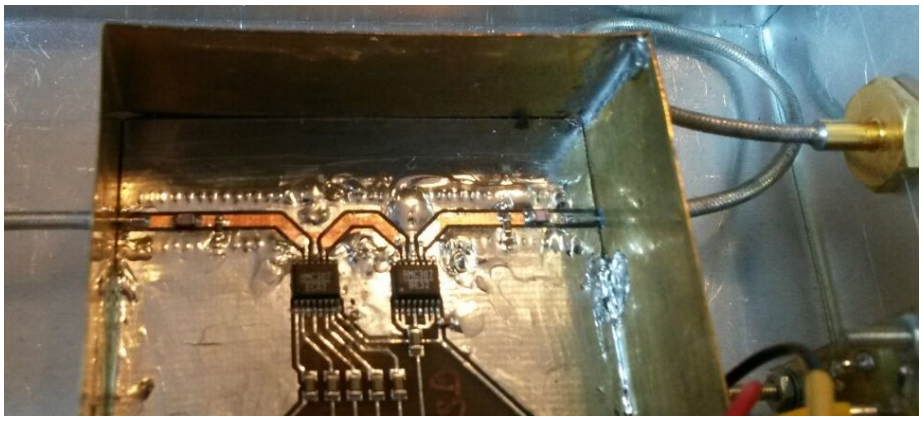
Slika 28: Električno vezje slabilca

Kljub temu, da ima uporabljeni mikroprocesor dovolj izhodov in bi lahko vseh šest stikalnih linj nastavljal direktno, sem se odločil, da preklopnike krmilim s pomočjo pomikalnega registra 74HC595. Pomikalni register od mikroprocesorja prejema vrednosti od 0 do 63 in jih pošilja na vhode S3, S2, S1 obeh preklopnikov. Za vrednosti nad 31 je potreben popravek v programski kodi za pravilno interpretacijo rezultata. Pri izhodni vrednosti 31 ali 32 je slabljenje še zmeraj 31 dB. Razlika je samo v tem, da pri vrednosti 31 slabljenje opravlja prvo integrirano vezje, pri vrednosti 32 pa drugo zaporedno vezano integrirano vezje, ki ima vsa notranja stikala stalno vklopljena na 31 dB slabljenja, ko je na vhodu logična ena.

Tudi slabilec je narejen na laminatu FR4, dvostransko, z maso spodaj in je vgrajen v medeninasto ohišje. Za prenos sintetiziranega signala sem naredil VF koplanarni vod, katerega dimenzije sem izračunal s pomočjo spletnega programa [14]. Širina linije znaša 1,27 mm (50 mil), razdalja do mase pa 0,51 mm (20 mil). Vzdolž linije sem s tanko žico prešival maso okoli linije na maso na drugi strani ploščice in jo nato pocinil.



Slika 29: Izdelava koplanarnega voda



Slika 30: Fotografija visokofrekvenčnega dela slabilca

Vstavitveno slabljenje slabilca se je po meritvi na vektorskem analizatorju vezij izkazalo za precej lepo premico, ki se kar dobro ujema s podatki v podatkovnem listu in je prikazano v spodnji tabeli.

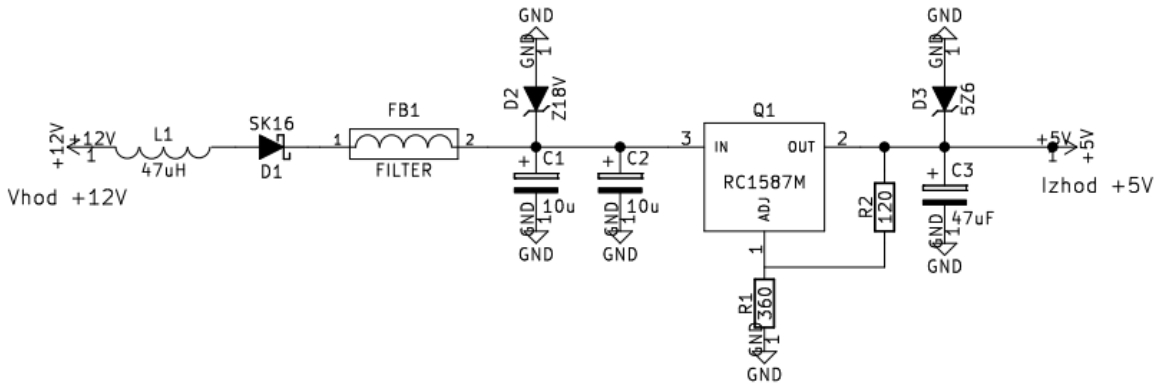
F(MHz)	a(dB)
45	-3,2
1000	-3,9
2000	-4,4
3000	-4,8
4400	-7,2

Tabela 2: Vstavitveno slabljenje slabilca

Slika 31: S21 slabilca

Napajalnik instrumenta

Generator in slabilec se napajata iz posebnega generatorja, mikrokontroler pa ima svoje lastno napajanje na razvojni plošči.



Slika 32: Napajalnik instrumenta

Celoten instrument potrebuje za normalno delovanje stabilno zunanje napajanje z napetostjo 12 V, ki zagotavlja vsaj 330 mA toka. Za napajanje je bil narejen napajalnik [16] z nastavljivim napetostnim regulatorjem RC1587M, ki zagotavlja dovolj velik napajali tok. Z uporoma R1 in R2 nastavimo izhodno napetost na 5V. Diode vezje ščitijo pred napačno polariteto, dušilki na vhodu pa dušijo morebitne moteče signale.

Krmiljenje

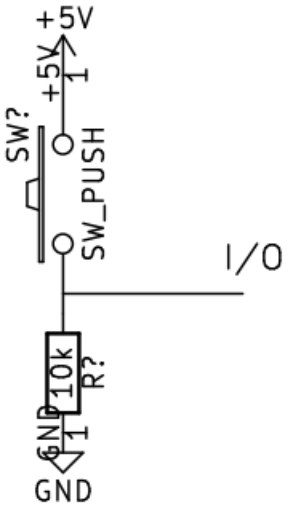
Instrument se krmili preko štirih tipk s pomočjo 16x2 LCD prikazovalnika HD44870. S tipkami levo in desno se pomikamo do spremenljivk, ki jih želimo nastavljati. S tipkami gor in dol nato povečujemo ali zmanjšujemo vrednost. Spreminjamo lahko frekvenco po vseh možnih cifrah. Izjema je le zadnja, ki predstavlja kHz. Tam je izbira odvisna izbranega frekvenčnega področja. Zaradi visoke primerjalne frekvence in omejenih FRAC in MOD registrov, ni mogoče generirati vseh frekvenc.

Poleg frekvence lahko v spodnji vrstici nastavljamo še slabljenje. S tipkami se premaknemo na desetice ali enice števila, ki izpisuje izhodno moč generiranega nosilca v dBm. S tipko navzdol vklapljamo slabilec po korakih 1 dB od 0 dB do 62 dB. Programska koda v mikrokrmilniku od kalibrirane izhodne moči odšteva nastavljeno slabljenje in izpisuje rezultat na zaslon. Desno od izpisa moči sem po vzoru kode Alain Forta [17] dodal zelo praktično kodo, s katero si shranimo 10 različnih frekvenc v spomin za hitrejše izbiranje. Če želimo prebrati kodo iz spomina, se s tipkami levo ali desno postavimo na številko spomina in s tipkami gor in dol izbiramo shranjene vrednosti. Če si želimo zapisati v spomin trenutno nastavljeno vrednost frekvence, se najprej postavimo na črko M. S tipko gor ali dol se črka M spremeni v črko Z, kar pomeni zapis. Nato se s tipkami levo ali desno pomaknemo na številko spomina in izberemo, na katero mesto si bomo zapisali. Ko sočasno

pritisnemo tipki gor in dol je frekvenca zapisana v spomin. Levo spodaj se nahaja še indikator zaklepa PLL zanke. Ko je vrednost L enaka 1 je PLL zanka zaklenjena. Vrednost se prebere iz MUX izhoda ADF4351, ki je sprogramiran za digitalen prikaz zaklepa zanke.

Slika 33: LCD prikaz

Tipke z 10 kΩ upori vezanimi na maso in LCD se napajajo iz razvojne plošče mikrokrmilnika Arduino Mega 2560. Vsi vhodi in izhodi iz mikrokrmilnika so oštevilčeni in poljubno nastavljivi v programski kodi, razen priključka za SPI vodilo CLK, št. 52 in DATA, št. 51. Celoten inštrument je vgrajen v aluminijasto ohišje priročne velikosti.



Slika 34: Vezava tipk

Opis delovanja bistvenih delov kode

Programska koda je napisana v programskem jeziku C. Pri pisanju sem se zgledoval po proizvajalčevi odprti kodi programa ADF435x [18] za testiranje njihove razvojne plošče z integriranim vezjem ADF4351, po prej omenjeni kodi [17] in arduino uporabniškim priročnikom [19].

Na začetku so definirane tri potrebne knjižnice, ki vsebujejo funkcije za SPI vodilo, LCD zaslon in funkcije za pisanje v flash pomnilnik mikroprocesorja.

```
#include <SPI.h>

#include <LiquidCrystal.h>

#include <EEPROM.h>
```

Sledijo nastavitve vhodnih in izhodnih priključkov, ter deklaracija spremenljivk, ki so v programu uporabljene.

Ena pomembnejših spremenljivk je polje `registers[6]` podatkovnega tipa `uint32_t`, ki vsebuje šest elementov v katerem so zapisane vrednosti šestih registrov ADF4351 sitentizatorja.

```
uint32_t registers[6] =  {0x338008, 0x8008041, 0x18004E42, 0x8004B3, 0x84003C, 0x580005};
```

V to polje so nastavljene začetne vrednosti registrov, ki jih nato nižje v programu spreminjamo po potrebi. Začetne vrednosti izračunamo s pomočjo podatkovnega lista, ali pa si pomagamo s proizvajalčevim grafičnim programom ADF435x za upravljanje njihove razvojne plošče.

Programi v okolju arduino so razdeljeni na dva dela. Prvi del je zapisan v funkciji `setup()` in se izvede samo enkrat po zagonu mikrokrmilnika. Namenjen je začetnim nastavitvam programa. Drugi del se imenuje `loop()` in predstavlja jedro kode, saj se ciklično izvaja dokler je mikrokrmilnik prižgan. Pred začetnimi nastavitvami se po navadi definira funkcije, ki se jih nato kliče v jedru kode.

Pomembnejši funkciji sta `WriteRegister32(const uint32_t vrednost)` in `SetADF4351()`.

```
void WriteRegister32(const uint32_t vrednost) {

        digitalWrite(LE, 0);

        for (int i = 3; i >= 0; i--)

        SPI.transfer((vrednost >> 8 * i) & 0xFF);

        digitalWrite(LE, 1);

        digitalWrite(LE, 0);

}
```

```
void SetADF4351() {

        for (int i = 5; i >= 0; i--)

        WriteRegister32(registers[i]);

}
```

Prva uporablja funkcijo `SPI.transfer` iz SPI.h knjižnice za prenos štiri krat po en bajt s pomočjo števca v for zanki. Argument vrednost se s pomočjo bitne operacij POMIK (>>) pomakne za 8 mest v desno in z bitno operaco IN (`& 0xFF`) ostane zadnjih 8 bitov za vsak `i` od 3 do 0. Pred pošiljanjem nastavimo zapah LE na nič. Ko se podatki preko SPI naložijo v medpomnilnik pomikalnega registra, pa sprostimo zapah z zapisom LE na ena in podatki se pošljejo naprej. S pomočjo funkcije `SetADF4351()` se tako zapiše vseh 6 registrov po zahtevanem vrstnem redu od R5 do R0. SPI vodilo se nastavi v `setup()` z ukazi:

```
SPI.begin();

SPI.setDataMode(SPI_MODE0);

SPI.setBitOrder(MSBFIRST);
```

Prvi ukaz nastavi SPI vodilo, drugi nastavi zajem podatkov na pozitivno urino stopnico (prehod iz nič na ena) in pošiljanje na negativno urino stopnico(prehod iz ena na nič), tretji pa določa pri katerem bitu se začnejo prenašat podatki (najvišjem MSB ali najnižjem LSB).

Na podoben način deluje funkcija za nastavljanje slabilca, le da tu ne uporablja strojnega SPI vodila ampak poljubno izbrane vhodno-izhodne priključke.

```
void SetATT() {

        b = att << 2;

        digitalWrite(LE_HMC, 0);

        shiftOut(DATA_HMC, CLK_HMC, LSBFIRST, b);

        digitalWrite(LE_HMC, 1);

        delay(10);

}
```

Podatki iz mikroprocesorja so speljani na pomikalni register integriranega vezja 74HC595. Slabljenje je zapisano v spremenljivki `att` in se po bitni operaciji POMIK za dva v levo zapiše v spremenljivko b. Pomikalni register ima osem izhodnih linij, za nastavljanje dveh HMC307 pa jih potrebujemo šest. S pomikom za dve mesti v levo in začetkom prenosa z LSB poskrbimo, da se ustrezni bit prenaša

vzporedno po ustrezni liniji, kot je to označeno v načrtu in se ujema s pravilnostno tabelo obeh slabilcev. Zapah nastavimo na logično nič in s funkcijo `shiftOut()`, ob spremembi ure na priključku CLK_HMC iz logične nič na logično ena, prenesemo podatke na liniji DATA_HMC v medpomnilnik pomikalnega registra. Ko se zapah sprosti, z nastavitvijo LE_HMC na 1, se podatki pošljejo na preklopnika 4053, ki nastavita slabilca. V tem primeru se prenos začne z najnižjim bitom (LSB).

Funkcija `izpisLCD ()` s pomočjo funkcije za branje tipk skrbi za interakcijo z uporabnikom, estetski izpis podatkov na LCD, preveri ali je prišlo od zaklepa zanke in popravi izpis izhodne moči za vrednosti slabljenja nad 31dB.

Funkcija `EEPROMWritelong(int naslov, long vrednost)` s pomočjo bitnih operacij IN in pomikanjem bitov, razstavi 32-bitno besedo na štiri 8-bitne in jih s pomočjo funkcije iz knjižnice EEPROM.h zapiše v flash pomnilnik mikroprocesorja. Za obratno operacijo skrbi funkcija `EEPROMReadlong(long naslov)`, ki prebere vsebino pomnilnika in jo nazaj sestavi v 32bitno besedo. Obe funkciji sta namenjeni samo shranjevanju frekvence v pomnilnik.

Začetna vrednost frekvence se v funkciji `setup()` prebere iz pomnilnika ali iz poljubno nastavljene spremenljivke `RFint`.

Izračun registrov za izbrano frekvenco

Jedro programa sestavljajo operacije za računanje in nastavljanje registrov sintetizatorja glede na izbrano frekvenco. Na začetku `loop()` funkcije imamo if zanko, ki preverja ali je prišlo do spremembe izbire frekvence. V tej zanki se izvedejo vsi pomembni izračuni in nastavitve registrov. Na začetku z drugo `if` zanko preverimo frekvenčno območje in na podlagi tega nastavimo vrednost deljenja izhodnega delilnika `OutputDivider`, ter zapišemo ustrezne vrednosti v register R4 s funkcijo `bitWrite()`.

```
..
if (RFout >= 2200) {
                OutputDivider = 1;
                OutputChannelSpacing=0.008; //korak bo 8kHz
                bitWrite (registers[4], 22, 0);
                bitWrite (registers[4], 21, 0);
                bitWrite (registers[4], 20, 0);
}
..
```

Prvi argument funkcije je številka registra, drugi predstavlja zaporedno številko bita, ki ga nastavljamo, tretji pa vrednost tega bita. Vrednosti izhodnega delilnika po območjih, lahko prepišemo kar iz programa za simulacijo ADIsimPLL. Vzporedno vrednosti izhodnega delilnika nastavljamo tudi ločljivost oz. frekvenčni korak, ki je na tem območju možen z izbrano primerjalno frekvenco 32 MHz in razpoložljivo velikostjo ulomkovnih registrov FRAC in MOD.

Za izračun registrov so uporabljene spodnje formule. Primerjalna frekvenca je nastavljena na 32. INT predstavlja celoštevilski del frekvence, FRAC in MOD pa ulomek oz. vrednost za decimalno piko. S pomočjo zgoraj nastavljenih vrednosti izhodnega delilnika in frekvenčnega koraka izračunamo imenovalec MODF in števec FRACF ulomka, ki nista celoštevilska. Po zaokrožitvi obeh vrednosti uporabimo funkcijo gcd, ki poišče največji skupni delitelj podanih argumentov in vrednost shranimo v spremenljivko dgcd. Imenovalec MOD nato z njim pokrajšamo in preverimo ali je manjši od 2, kar je njegova najmanjša možna vrednost. Sledi ponovni izračun pokrajšane vrednosti števca FRACF in njegovo zaokroževanje na celoštevilsko vrednost.

```
PFD = 32;

INT = (RFout * OutputDivider) / PFD;

MODF = (PFD / OutputChannelSpacing);

FRACF = (((RFout * OutputDivider) / PFD) - INT) * MODF;

FRAC = round(FRACF);

MOD = round(MODF);


dgcd = gcd(MOD, FRAC);

MOD /= dgcd;

if (MOD==1) MOD=2;


FRACF = (((RFout * OutputDivider) / PFD) - INT) * MOD;

FRAC = round(FRACF);
```

Tako izračunane vrednosti INT, FRAC in MOD so primerne za vpis v ustrezne registre po navodilih iz podatkovnega lista. Vrednosti so zapisane z bitnimi operacijami (pomikanjem bitov levo ali desno) in funkcijo `bitWrite()`.

Ostale nastavitve

V programu sledi še nekaj nastavitev registrov (koda v prilogi). Med pomembnejše spada nastavitev toka črpalke nabojev s katero lahko spreminjamo tudi dinamiko zančnega sita.

Ker imamo najvišjo možno primerjalno frekvenco 32 MHz, je potrebno povečati frekvenco izbire VCO področja. Frekvenca je nastavljena na 250 kHz. Manjše vrednosti lahko povzročijo, da se zanka ne zaklene. Maksimalna možna vrednost je 500 kHz.

Integrirano vezje lahko deluje v celoštevilskem ali ulomkovnem načinu, zato je potrebno preveriti tudi način delovanja in primerno načinu delovanja nastaviti štiri registre. Dva sta namenjena detekciji zaklepa PLL zanke (angl. lock detect function in angl. lock detect precision), dva pa odpravljanju neželjenih špic, ki jih lahko povzroča črpalka nabojev (angl. antibackslash pulse width in angl. charge pump cancelation) in sta nastavljena po navodilih proizvajalca.

Nastavitev vrednost preddelilnika vpliva samo na velikost registra INT in je nastavljena na 8/9, kar naj bi po navodilih iz podatkovnega lista omejilo vrednost registra INT na minimalno 75. V tem primeru naj ne bi bilo mogoče izbirati frekvenc pod 37,5 MHz. V preizkusu se je izkazalo, da je vseeno možno generirati frekvence vse do 34,375 MHz, kjer je vrednost INT = 68.

Izhodna moč je nastavljena na maksimalno +5 dBm in VF izhod je ugasnjen, dokler PLL zanka ni zaklenjena (angl. Mute until Lock Detect – MTLD). Po priporočilu iz podatkovnega lista, je fazna beseda nastavljena na 1.

V kodo je možno dodati nastavitve še za druge registre, a ti za uporabo instrumenta kot generator signala niso bistvenega pomena. Sintetizator je prvinsko namenjen uporabi v radijskih komunikacijskih napravah.

Na koncu se nahaja preprost a obsežen del kode, ki tipkam dodaja funkcijo, da lahko nastavljamo spremenljivke, ki jih vidimo na LCD zaslonu. Med drugim, koda tu popravlja celoštevilsko RFint spremenljivko z zaokroževanjem (z deljenjem in množenjem) tako, da uporabnik ne more izbrati frekvence, ki na določenem frekvenčnem področju ni mogoča.

```
if ((RFint >= 2200000) && (RFint < 4400000)) {

                                RFint = RFint + 8; //korak je 8kHz

                                za = RFint/8;

                                RFint = za * 8;

                        }
```

Ostali deli kode so opisani v komentarjih kode.

V prilogo sem dodal še krajši del kode, ki je primeren za preizkušanje delovanja ADF4351. Testna koda deluje enako kot glavna koda, le da nima nobenih drugih funkcij zraven. Vse uporabljene funkcije so zgoraj že opisane. V polje `registers[6]` vnesemo poljubno vsebino registrov in jo pošljemo prek SPI vodila na integrirano vezje.

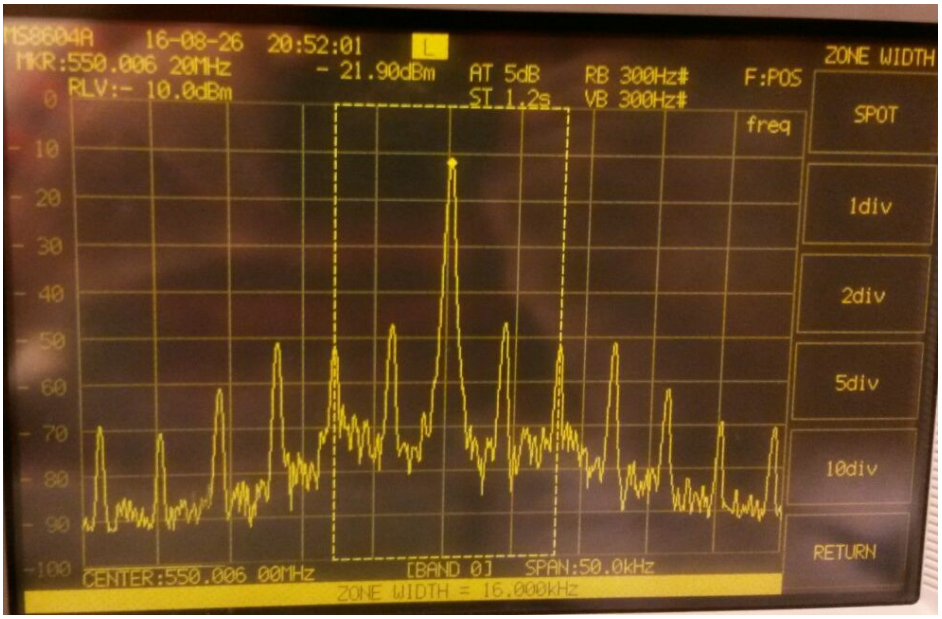
Zaključne besede

Po dosedanjih testiranjih koda deluje po predvidevanjih. V kodo je vgrajena še možnost kontrole izračunanih registrov preko USB priključka s poljubnim terminalskim programom. Hitrost vodila je v programu nastavljena na 115200 bit/s. Pravilnost izračunanih registrov lahko nato preverimo s proizvajalčevim programom ADF435x. Praktično bi bilo v kodo vgraditi še možnost, da se v spomin shranjuje tudi nastavljeno slabljenje pri določeni frekvenci in izboljšati odziv pri krmiljenju z več tipkami sočasno.

Omejitve ADF4351

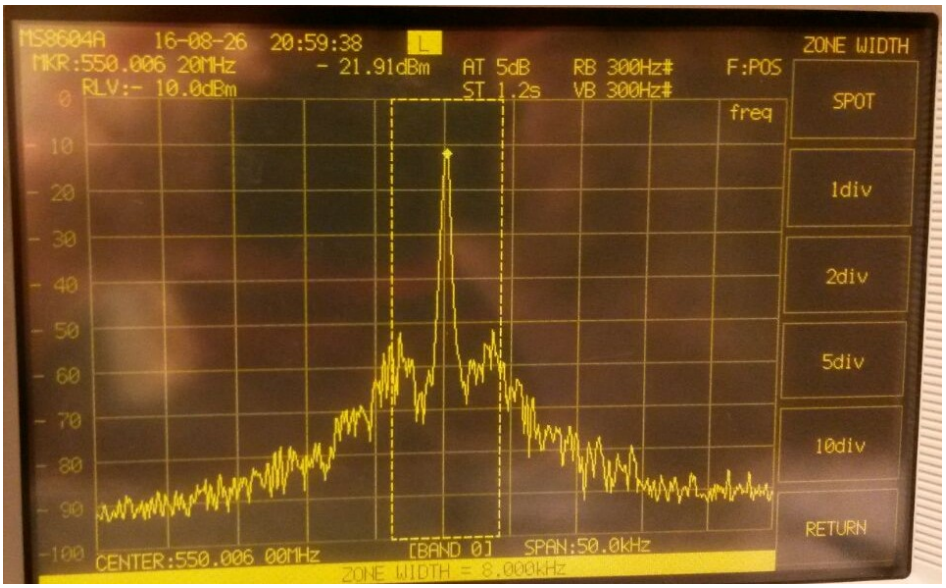
Integrirano vezje ADF4351 ima ulomkovna registra FRAC in MOD omejena na velikost 12 bitov, kar na frekvenčnih področjih nad 550 MHz pri najvišji možni primerjalni frekvenci 32 MHz onemogoča sintezo frekvenc s korakom 1 kHz. Sinteza je tako možna edino z izbiro druge primerjalne frekvence, kar zahteva novo načrtovanje zančnega sita.

Drugi problem integriranega vezja predstavljajo nezaželene harmonske špice, ki jih povzroča ulomkovni interpolator pri določenih frekvencah. Na spodnji sliki vidimo primer pri 550,006 MHz. Špice se ponavljajo na intervalu $\frac{f_{PFD}}{L}$, kjer je v števcu primerjalna frekvenca v imenovalcu pa dolžina ponavljajočega kodnega zaporedja v Σ-Δ modulatorju. V konkretnem primeru je MOD = 4000, kar po enačbi iz tabele 7 v podatkovnem listu [1] prinese moteče špice na vsakih 32 MHz / (2 * 4000) = 0,004 MHz.



Slika 35: Spekter nosilca pri 550,006MHz

S spremembo primerjalne frekvence na polovico, moteče špice izginejo.



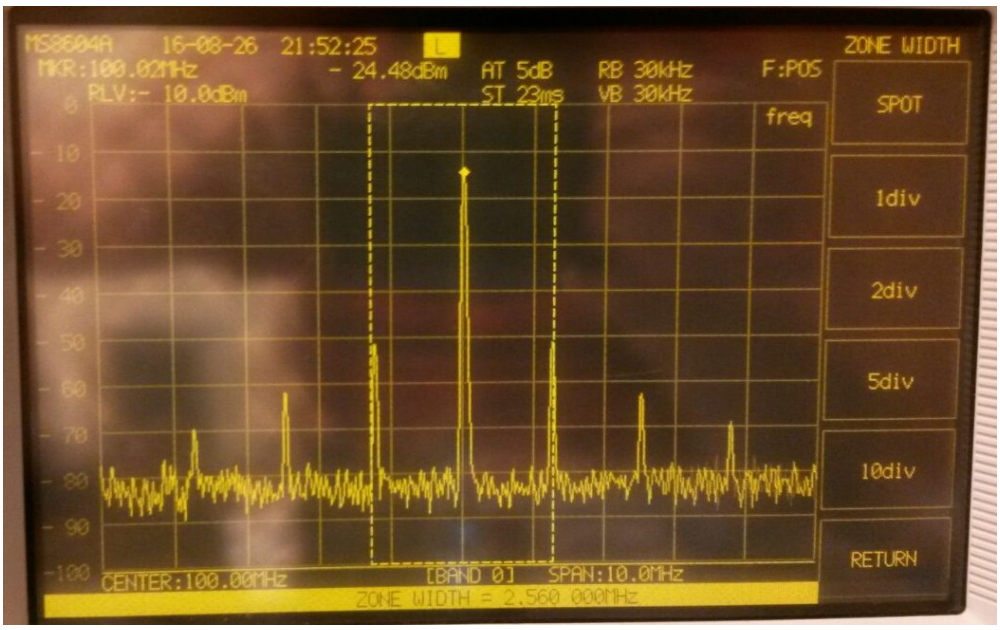
Slika 36: Spekter nosilca pri 550,006MHz in polovični primerjalni frekvenci

Ker je možno primerjalno frekvenco s programsko kodo spreminjati, bi bilo dobro preveriti ali je smiselno v kodo vgraditi popravljanje primerjalne frekvence za takšne primere na celotnem območju delovanja sintetizatorja. Potrebno bi bilo preveriti ali se zanka PLL zaklepa in ali je s popravljanjem tokovne črpalke nabojev možno tudi primerno popravljanje zančnega sita ob nezaklepanju zanke.

Za detekcijo PLL zaklepa programska koda preverja stanje nastavljivega MUX izhoda. Poleg tega bi morda bilo praktično speljati še LED diodo iz priključka 25 na ADF4351 izven ohišja instrumenta.

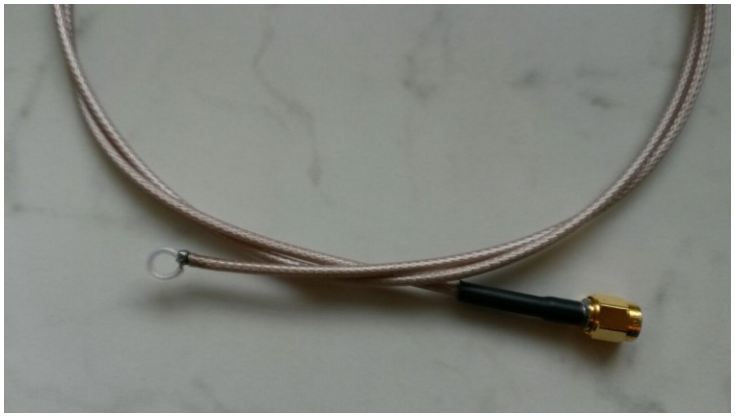
Primer slabega napajanja

Pri sestavljanju enega od dveh generatorjev sem po nekem naključju vgradil primer ADP150 regulatorja, ki ni deloval pravilno. Na izhodnem spektru so se pojavljale visoke moteče špice v presledkih okoli 1,2 MHz.



Slika 37: Spekter motenj slabega napajanja

Napajanje VCO-ja je zelo pomembno za izhodni spekter in mora imeti čim manjši šum. Pri odkrivanju izvora motenj v vezju je zelo praktično orodje induktivna sonda z zanko, s katero lahko potipamo po vezju in preverimo od kje prihajajo motnje. S pomočjo sonde sem odkril izvor motenj in zamenjal regulator.



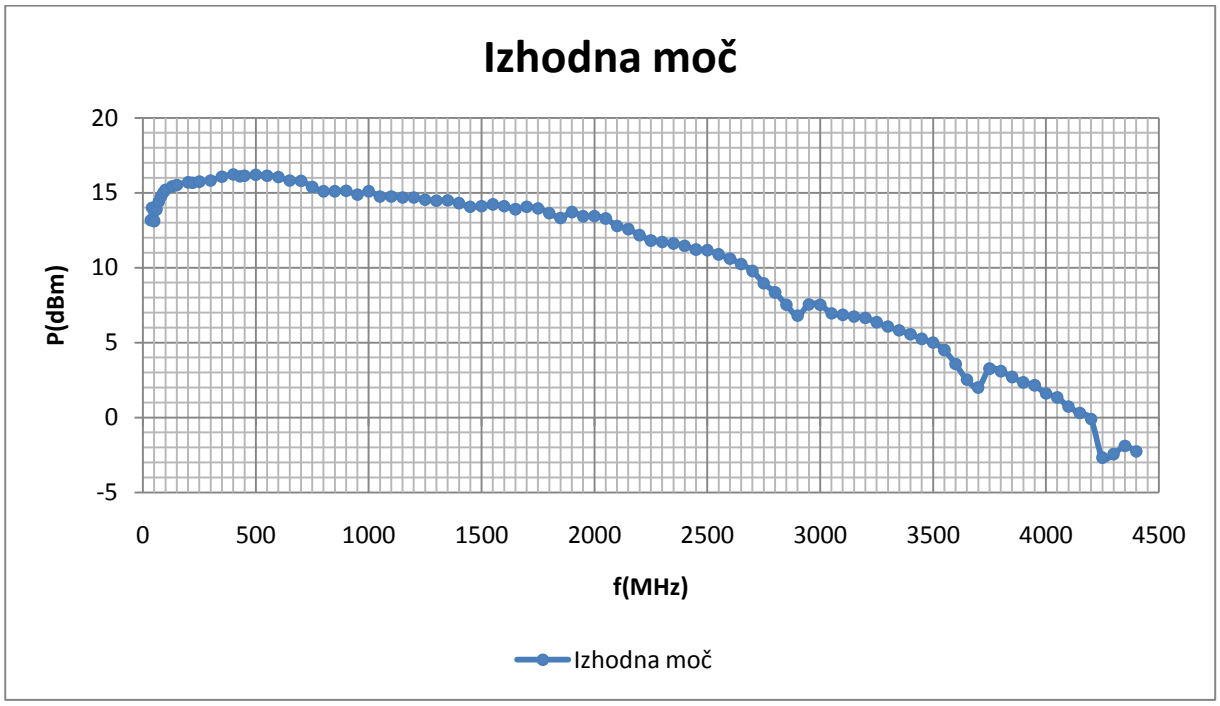
Slika 38: Induktivna sonda z zanko

Izhodni signal

Integrirano vezje ADF4351 ima izhodni delilnik z digitalno logiko, ki povzroča pravokoten izhodni signal in harmonske večkratnike generiranega nosilca. Uporabnik se mora tega dejstva zavedati in po potrebi poskrbeti za ustrezna izhodna sita. Velikost drugih in tretjih harmonskih komponent se približno ujema s podatki na podatkovnem listu (-20 dBc in -10 dBc). Velikost špic, ki jih povzroča referenčni vir TCXO pri 32 MHz pa znašajo med -90 dBc in -80 dBc.

Na spodnji sliki je prikazan graf moči generiranega nosilca pri slabljenju 0 dB v zadnji verziji instrumenta. Na grafu sta vidni območji slabljenja med 2,5 GHz in 4 GHz, ki sta verjetno posledica vezave priključnih sponk baluna na vezje, saj se kar lepo ujemata z grafom na sliki 21. Če graf primerjamo s primerjalnim grafom prvega testnega vezja sintetizatorja na sliki 24. (temno modra črta, izhod iz sintetizatorja pri +5 dBm preko baluna in ojačevalnika) in mu prištejemo vstavitveno slabljenje slabilca, se grafi približno ujemajo.

Izdelava inštrumenta zahteva natančnost pri delu in veliko ročnih spretnosti. Že povsem majhna napaka pri slabem cinjenju kondenzatorja lahko povzroči izgubo par dBm signala na izhodu. Vse komponente od vezja do ohišja so bile narejene ročno.



Slika 39: Graf izhodne moči

Ne glede na zgoraj opisane pomankljivosti je nastal praktično povsem uporaben radio-frekvenčni vir s slabilcem, ki deluje na območju od 34,375 MHz do 4400 MHz in slabi po korakih 1 dB od 0 dB do 62 dB. S pomočjo programske kode je mogoče sintetizirani signal preprosto umeriti, da prikazuje točno izhodno moč nosilca. Instrument je lahko tudi osnova za razvoj boljšega generatorja z zmogljivejšim sintetizatorjem ali drugih naprav, ki delujejo na osnovi sinteze signala s fazno sklenjeno zanko.

Priloge

Programska koda

```
/*************************************************************************/
/*** RF GENERATOR, koda za arduino (Mega 2560) v201609010440 by S57NZI ***/
/*************************************************************************/

//potrebujemo funkcije iz spodnjih knjiznic za SPI prenos, prikaz na LCD in zapis v flash pomnilnik
#include <SPI.h>
#include <LiquidCrystal.h>
#include <EEPROM.h>

//izberi stevilko pina na arduinotu za komunikacijo z ADF4351
//SPI vodilo uporablja se pin 52 za CLK in 51 za DATA, ki jih ni potrebno posebej definirati
#define LE 47
#define MUX 46
#define PDRF 48
#define CE 49

//pini za pomikalni register 74HC595
#define LE_HMC 40
#define CLK_HMC 42
#define DATA_HMC 43

//pini za tipke
#define DESNA 35
#define GOR 37
#define DOL 34
#define LEVA 36
#define IZBERI 24 //tu pin ni uporabljen, sluzi samo kot vrednost pri uporabi dveh tipk hkrati

//stevilke pinov kamor priklopimo pine hd44780 LCD (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
byte pcurs = 8; //pcurs predstavlja položaj kurzorja na zaslonu 0-15 mest
byte line = 0; // zgornja/spodnja vrstica 0/1 na zaslonu
```

```
byte eprom; // številka spomina v flashu arduinota


//zacetne nastavitve vseh sestih registrov s pomocjo orodja proizvajalca, spodaj naprej jih lahko
poljubno nastavljamo

uint32_t registers[6] =  {0x338008, 0x8008041, 0x18004E42, 0x8004B3, 0x84003C, 0x580005};


int ZEE=0; //spremenljivka za zapis v spomin

int tipka = 0; // vmesna spremenjljivka za izbrano tipko

byte b, att, atts;  //spremenljivke za nastavljanje slabljena

int moc; //spremenljivka za izpis moci na zaslon


//spremenljivke za izracun in izpis frekvence

double RFout, OutputChannelSpacing, FRACF, VCO, MODF;

unsigned int long INT, FRAC, MOD; //celostevilske spremenjlivke za vpis v registre


unsigned int long RFint, RFints, RFcalc; // nova in stara, ter pomozna  spremenljivka za izpis
frekvence na LCD

unsigned int long za; //spremenljivka za zaokrozevanje na frekvenco, ki je mozna na danem obmocju (1,
2, 4, 8 kHz korak)

unsigned int dgcd;  // spremenljivka za najvecji skupni delitelj


//spremenjivka za primerjalno frekvenco, ki je trenutno enaka TCXO v MHz (default R=1)

unsigned int long PFD = 32;


byte OutputDivider; //delilnik na izhodu(1-2-4-8-16-32-64)


//funkcija za branje tipk

int beri_tipke() {

        if ((digitalRead(GOR) == 1) && (digitalRead(DOL) == 1)) {

                delay(250);

                return IZBERI;

                delay(100);

        }

        if (digitalRead(DESNA) == 1) {

                delay(250);

                return DESNA;

        }

        if (digitalRead(GOR) == 1) {
```

```
                delay(250);
                return GOR;
        }
        if (digitalRead(DOL) == 1) {
                delay(250);
                return DOL;
        }
        if (digitalRead(LEVA) == 1) {
                delay(250);
                return LEVA;
        }
        if (digitalRead(IZBERI) == 1) {
                delay(250);
                return IZBERI;
        }
}
//primitivna kalibracija za izhodno moc
//f2m() izpise zaokrozeno moc nosilca v dBm glede na izbrano frekvencno obmocje
int f2m (unsigned int f) {
        if(f < 40) return 13;
        if(f >= 40 && f < 50 ) return 13;
        if(f >= 50 && f < 60) return 14;
        if(f >= 60 && f < 70 ) return 14;
        if(f >= 70 && f < 80 ) return 15;
        if(f >= 80 && f < 90 ) return 15;
        if(f >= 90 && f < 100 ) return 15;
        if(f >= 100 && f < 130 ) return 15;
        if(f >= 130 && f < 150 ) return 15;
        if(f >= 150 && f < 200 ) return 16;
        if(f >= 200 && f < 220 ) return 16;
        if(f >= 220 && f < 250 ) return 16;
        if(f >= 250 && f < 300 ) return 16;
        if(f >= 300 && f < 350 ) return 16;
        if(f >= 350 && f < 400 ) return 16;
        if(f >= 400 && f < 430 ) return 16;
        if(f >= 430 && f < 450 ) return 16;
```

```
if(f >= 450 && f < 500 ) return 16;
if(f >= 500 && f < 550 ) return 16;
if(f >= 550 && f < 600 ) return 16;
if(f >= 600 && f < 650 ) return 16;
if(f >= 650 && f < 700 ) return 16;
if(f >= 700 && f < 750 ) return 16;
if(f >= 750 && f < 800 ) return 15;
if(f >= 800 && f < 850 ) return 15;
if(f >= 850 && f < 900 ) return 15;
if(f >= 900 && f < 950 ) return 15;
if(f >= 950 && f < 1000 ) return 15;
if(f >= 1000 && f < 1050 ) return 15;
if(f >= 1050 && f < 1100 ) return 15;
if(f >= 1100 && f < 1150 ) return 15;
if(f >= 1150 && f < 1200 ) return 15;
if(f >= 1200 && f < 1250 ) return 15;
if(f >= 1250 && f < 1300 ) return 14;
if(f >= 1300 && f < 1350 ) return 14;
if(f >= 1350 && f < 1400 ) return 14;
if(f >= 1400 && f < 1450 ) return 14;
if(f >= 1450 && f < 1500 ) return 14;
if(f >= 1500 && f < 1550 ) return 14;
if(f >= 1550 && f < 1600 ) return 14;
if(f >= 1600 && f < 1650 ) return 14;
if(f >= 1650 && f < 1700 ) return 14;
if(f >= 1700 && f < 1750 ) return 14;
if(f >= 1750 && f < 1800 ) return 14;
if(f >= 1800 && f < 1850 ) return 13;
if(f >= 1850 && f < 1900 ) return 14;
if(f >= 1900 && f < 1950 ) return 14;
if(f >= 1950 && f < 2000 ) return 13;
if(f >= 2000 && f < 2050 ) return 13;
if(f >= 2050 && f < 2100 ) return 13;
if(f >= 2100 && f < 2150 ) return 13;
if(f >= 2150 && f < 2200 ) return 12;
if(f >= 2200 && f < 2250 ) return 12;
```

```
if(f >= 2250 && f < 2300 ) return 12;
if(f >= 2300 && f < 2350 ) return 12;
if(f >= 2350 && f < 2400 ) return 12;
if(f >= 2400 && f < 2450 ) return 11;
if(f >= 2450 && f < 2500 ) return 11;
if(f >= 2500 && f < 2550 ) return 11;
if(f >= 2550 && f < 2600 ) return 11;
if(f >= 2600 && f < 2650 ) return 10;
if(f >= 2650 && f < 2700 ) return 10;
if(f >= 2700 && f < 2750 ) return 9;
if(f >= 2750 && f < 2800 ) return 9;
if(f >= 2800 && f < 2850 ) return 8;
if(f >= 2850 && f < 2900 ) return 7;
if(f >= 2900 && f < 2950 ) return 7;
if(f >= 2950 && f < 3000 ) return 8;
if(f >= 3000 && f < 3050 ) return 7;
if(f >= 3050 && f < 3100 ) return 7;
if(f >= 3100 && f < 3150 ) return 7;
if(f >= 3150 && f < 3200 ) return 7;
if(f >= 3200 && f < 3250 ) return 7;
if(f >= 3250 && f < 3300 ) return 6;
if(f >= 3300 && f < 3350 ) return 6;
if(f >= 3350 && f < 3400 ) return 6;
if(f >= 3400 && f < 3450 ) return 5;
if(f >= 3450 && f < 3500 ) return 5;
if(f >= 3500 && f < 3550 ) return 4;
if(f >= 3550 && f < 3600 ) return 4;
if(f >= 3600 && f < 3650 ) return 4;
if(f >= 3650 && f < 3700 ) return 3;
if(f >= 3700 && f < 3750 ) return 3;
if(f >= 3750 && f < 3800 ) return 3;
if(f >= 3800 && f < 3850 ) return 3;
if(f >= 3850 && f < 3900 ) return 3;
if(f >= 3900 && f < 3950 ) return 2;
if(f >= 3950 && f < 4000 ) return 2;
if(f >= 4000 && f < 4050 ) return 1;
```

```
        if(f >= 4050 && f < 4100 ) return 1;
        if(f >= 4100 && f < 4150 ) return 1;
        if(f >= 4150 && f < 4200 ) return 0;
        if(f >= 4200 && f < 4250 ) return -1;
        if(f >= 4250 && f < 4300 ) return -3;
        if(f >= 4300 && f < 4350 ) return -2;
        if(f >= 4350 && f <= 4400 ) return -2;
}

// funkcija za izpis na LCD zaslon
void izpisLCD () {
//prva vrstica
        lcd.setCursor(0, 0); //postavi kurzor na prvo mesto v prvo vrstico
        lcd.print("Frek:");

//popravi izpis presledeka/stevilk pri razlicni dolzini stevilk frekvence
        if (RFint < 1000000) lcd.print(" ");
        if (RFint < 100000) lcd.print(" ");
        if (RFint < 10000) lcd.print(" ");

        lcd.print(RFint / 1000);  // izpisemo del pred decimalno piko
        lcd.print(".");
//popravi zapis za decimalno piko
        RFcalc = RFint - ((RFint / 1000) * 1000);
        if (RFcalc < 10) lcd.print("0");
        if (RFcalc < 100) lcd.print("0");
        lcd.print(RFcalc);
        lcd.print("MHz");
//druga vrstica
        lcd.setCursor(0,1);
        lcd.print("P=");  // izpisemo moc
        if ((moc < 10) && (moc >= 0)) {
               lcd.print(" +");
        }
        if (moc > 9) {
                lcd.print("+");
```

```
        }
        if ((moc > -10) && (moc < 0)) {
                lcd.print(" ");
        }
//popravek izpisa moci ko se prizge drugi HMC307
        if((att>=32) && (att<=63)) {
                 lcd.print(moc+1, DEC);
        }
        else {
                lcd.print(moc, DEC);
        }
        lcd.print("dBm");
//izpis za spominska mesta
        if (ZEE == 0) {
                lcd.print(" M="); //izpisemo M, ko beremo spomin
        }
        else lcd.print(" Z="); // izpisemo Z ko zapisujemo v spomin
        lcd.print(eprom,DEC);
//izpis za lock detect kontrolo, da vemo, ce je fazna zanka ujeta
        if ((digitalRead(MUX)==1)) {
                lcd.print(" L:1");
        }
        else lcd.print(" L:0");

        lcd.setCursor(pcurs,line); //postavi kurzor v stolpec, vrstico
}

//funkicja za prenos preko SPI
void WriteRegister32(const uint32_t vrednost) {
        digitalWrite(LE, 0); // LE nastavimo zapah (load enable) na 0
        for (int i = 3; i >= 0; i--) // stevec i, za 4 x 1 bajt (od cetrtega do prvega)
//SPI.transfer prenasa po en byte v oklepaju, vrednost pomikamo za 8 bitov v desno,
//mnozimo s stevcem i in izvajamo & operacijo z 11111111 (ostane zadnjih 8 bitov)
        SPI.transfer((vrednost >> 8 * i) & 0xFF);
//sprostimo zapah, posljemo podatke, LE nastavimo na 1 in nato spet na 0
        digitalWrite(LE, 1);
```

```
        digitalWrite(LE, 0);
}

void SetADF4351() {
        for (int i = 5; i >= 0; i--)
//klicemo zgornjo funkcijo po stevcu i od 5 do 0 za zapis vseh sestih registrov v vrstnem redu R5->R0,
kot zateva ADF4351
        WriteRegister32(registers[i]);
}

//funkcija nastavi slabljenje na 2xHMC307 preko pomikalnega registra 74HC595
//tu uporabljamo druge pine in drugo funkcijo za prenasanje
void SetATT() {
        b = att << 2;  // imamo samo 6 bitov (0-63), dva nepotrebna bita odrezemo
        digitalWrite(LE_HMC, 0); //nastavimo zapah na 0
// funkcija za pomik b bitov, zacnemo prenasat prvo najnizji bit
        shiftOut(DATA_HMC, CLK_HMC, LSBFIRST, b);
        digitalWrite(LE_HMC, 1); // sprostimo zapah, podatki, se pošljejo
        delay(10);
}

// funkcija za vpis v EEPROM, da si shranimo frekvenco
void EEPROMWritelong(int naslov, long vrednost)
      {
//s pomikanjem bitov na desno in and operacijo (rezanjem odvecnih bitov) razstavimo 32bit besedo na 4
byte
      byte cetrti = (vrednost & 0xFF);
      byte tretji = ((vrednost >> 8) & 0xFF);
      byte drugi = ((vrednost >> 16) & 0xFF);
      byte prvi = ((vrednost >> 24) & 0xFF);
//vsak byte zapisemo na svoj naslov po vrsti
      EEPROM.write(naslov, cetrti);
      EEPROM.write(naslov + 1, tretji);
      EEPROM.write(naslov + 2, drugi);
      EEPROM.write(naslov + 3, prvi);
      }
```

```
//obratna operacija od prejsnje funkcije

//preberemo zapisane stiri bajte v spominu in jih s pomikanjem v levo in and operacijo z enicami
sestejemo v 32bit besedo

long EEPROMReadlong(long naslov) {

      long cetrti = EEPROM.read(naslov);

      long tretji = EEPROM.read(naslov + 1);

      long drugi = EEPROM.read(naslov + 2);

      long prvi = EEPROM.read(naslov + 3);

      return ((cetrti << 0) & 0xFF) + ((tretji << 8) & 0xFFFF) + ((drugi << 16) & 0xFFFFFF) + ((prvi <<
24) & 0xFFFFFFFF);

}



//funkcija poisce najvecji skupni delitelj dveh podanih vrednosti, za pokrajsat ulomek FRAC/MOD

int gcd(uint32_t u, uint32_t v) {

        uint32_t t;

        while (v) {

                t = u;

                u = v;

                v = t % v; //ostanek po deljenju (modulo)

        }

        return u ;

}



//funkcija, ki se zažene samo enkrat, pri vklopu

void setup() {



//nastavi vhode/izhode na zgoraj definiranih pinih

//tipke

        pinMode(DESNA, INPUT);

        pinMode(GOR, INPUT);

        pinMode(DOL, INPUT);

        pinMode(LEVA, INPUT);

//ADF4351

        pinMode(MUX, INPUT);

        pinMode(LE, OUTPUT);

        pinMode(PDRF, OUTPUT);

//vklopi cip in RF izhod
```

```
        digitalWrite(CE, 1); //chip enable
        digitalWrite(PDRF, 1); //RFout enable
//HMC307
        pinMode(LE_HMC, OUTPUT);
        pinMode(CLK_HMC, OUTPUT);
        pinMode(DATA_HMC, OUTPUT);

        lcd.begin(16, 2); //2x16 display
        lcd.display();

        Serial.begin (115200); //nastavi hitrost na 115200 za serijsko povezavo
//uvodni izpis na LCD
        lcd.print("RFGEN 35-4400MHz");
        lcd.setCursor(0, 1);
        lcd.print("   ATT 0-62dB   ");
        pcurs = 8; line = 0;
        delay(2000);
//SPI nastavitve
        digitalWrite(LE, 1);                  // postavi LE na 1
        SPI.begin();                          // Init SPI vodilo
        SPI.setDataMode(SPI_MODE0);           // CPHA = 0 nastavi polariteto in fazo ure
        SPI.setBitOrder(MSBFIRST);            // MSB gre prvi naprej
//preberi frekvenco iz spomina, ce je vpisana, drugace nastavi na poljubno 88MHz
        if (EEPROM.read(111)==77) {           //kontrola, ce je bilo ze kdaj vpisano v flash (77 na
naslovu 111), poljubno izbrano
                RFint=EEPROMReadlong(eprom*4);
        }
        else {
                RFint=88000; //prikazi 88MHz
        }

        RFints = 12345;// da bo RFints razlicen od RFout na zacetku
        RFout = RFint / 1000 ; // izhodna frekvenca (v MHz)
        ZEE=0;
        lcd.blink();
        izpisLCD();
        delay(500);
```

```
        moc = f2m(RFout); //izpisi moc, glede na trenutno frekvenco
        att=0;
        SetATT(); //zacetno slabljenje je 0dB
}

//tu se zacne zanka, ki se ponavlja, dokler je MCU prizgan
void loop() {
        RFout = RFint;
        RFout = RFout/1000; //dobimo vrednost v MHz
        if (RFint != RFints) { //pogoj, ce se zgodi sprememba frekvence
//nastavi output divider(1, 2, 4, 8, 16, 32, 64) in channel spacing, glede na frekvencno obmocje
//PFD bo nastavljen na 32MHz in channel spacing je lahko 8 ali 4 ali 2 ali 1kHz, odvisno od
frekvencnega obmocja,
//zaradi premajhnih FRAC/MOD registrov, drugace bi morali spreminjati primerjalno frekvenco

                if (RFout >= 2200) {
                        OutputDivider = 1;
                        OutputChannelSpacing=0.008;
                        bitWrite (registers[4], 22, 0);
                        bitWrite (registers[4], 21, 0);
                        bitWrite (registers[4], 20, 0);
                }
                if (RFout < 2200) {
                        OutputDivider = 2;
                        OutputChannelSpacing=0.004;
                        bitWrite (registers[4], 22, 0);
                        bitWrite (registers[4], 21, 0);
                        bitWrite (registers[4], 20, 1);
                }
                if (RFout < 1100) {
                        OutputDivider = 4;
                        OutputChannelSpacing=0.002;
                        bitWrite (registers[4], 22, 0);
                        bitWrite (registers[4], 21, 1);
                        bitWrite (registers[4], 20, 0);
                }
                if (RFout < 550)  {
```

```
                        OutputDivider = 8;
                        OutputChannelSpacing=0.001;
                        bitWrite (registers[4], 22, 0);
                bitWrite (registers[4], 21, 1);
                bitWrite (registers[4], 20, 1);
                }
                if (RFout < 275)  {
                        OutputDivider = 16;
                        OutputChannelSpacing=0.001;
                        bitWrite (registers[4], 22, 1);
                        bitWrite (registers[4], 21, 0);
                        bitWrite (registers[4], 20, 0);
                }
                if (RFout < 137.5) {
                        OutputDivider = 32;
                        OutputChannelSpacing=0.001;
                        bitWrite (registers[4], 22, 1);
                        bitWrite (registers[4], 21, 0);
                        bitWrite (registers[4], 20, 1);
                }
                if (RFout < 68.75) {
                OutputDivider = 64;
                        OutputChannelSpacing=0.001;
                        bitWrite (registers[4], 22, 1);
                        bitWrite (registers[4], 21, 1);
                        bitWrite (registers[4], 20, 0);
                }

//formule za izracun INT, FRAC, MOD za vpis v registre
                PFD = 32;
                INT = (RFout * OutputDivider) / PFD;
                MODF = (PFD / OutputChannelSpacing);
                FRACF = (((RFout * OutputDivider) / PFD) - INT) * MODF;
                FRAC = round(FRACF); // zaokrozimo na celi del
                MOD = round(MODF);
// frekvenca VCO, samo za informativni izpis preko serijskega porta, drugace ni v uporabi
```

```
                VCO = (INT+(FRACF/MODF)) * PFD;

//pokrajsaj ulomek FRAC/MOD

                dgcd = gcd(MOD, FRAC);

                MOD /= dgcd;  //MOD delimo z najvecjim skupnim deljiteljem

                if (MOD==1) MOD=2; //ne sme biti manj od 2


//ponovno izracunamo FRAC s pokrajsanim MOD

                FRACF = (((RFout * OutputDivider) / PFD) - INT) * MOD;

                FRAC = round(FRACF);


//izpis nekaterih spremenljivk preko serijske konzole, npr. za kotrolo s pomocjo proizvajalcevega
testnega programa

Serial.print("RFout: ");

Serial.print(RFout,DEC);

Serial.print("\r\n");


Serial.print("VCO: ");

Serial.print(VCO,DEC);

Serial.print("\r\n");


Serial.print("PFD: ");

Serial.print(PFD,DEC);

Serial.print("\r\n");


Serial.print("GCD: ");

Serial.print(dgcd,DEC);

Serial.print("\r\n");


Serial.print("INT: ");

Serial.print(INT,DEC);

Serial.print("\r\n");

Serial.print(INT,BIN);

Serial.print("\r\n");


Serial.print("FRAC: ");

Serial.print(FRAC, DEC);

Serial.print("\r\n");
```

```
Serial.print(FRAC, BIN);

Serial.print("\r\n");


Serial.print("MOD: ");

Serial.print(MOD,DEC);

Serial.print("\r\n");

Serial.print(MOD,BIN);

Serial.print("\r\n");


Serial.print("OutputDivider: ");

Serial.print(OutputDivider, DEC);

Serial.print("\r\n");


Serial.print("OutputChannelSpacing: ");

Serial.print(OutputChannelSpacing, DEC);

Serial.print("\r\n\n");


//izracunane vrednosti pripravimo za vpis v registre cipa, po navodilih v podatkovnem listu

//v register R0 bomo zapisali vrednost INT in FRAC

                registers[0] = 0;

                registers[0] = INT << 15; // pomik za 15 v levo, na desni dodamo vrednosti INT 15 nicel

                FRAC = FRAC << 3; // pomik za 3 v levo, FRAC na desni dodamo 3 nicle

                registers[0] = registers[0] + FRAC; // pristejemo FRAC

//MOD vpisemo na ustrezno mesto v register R1

                registers[1] = 0;

                registers[1] = MOD << 3; // dodaj 3 nicle na desni

                registers[1] = registers[1] + 1 ; //postavi zadnji bit na 1 (kontrolni bit)

                bitWrite (registers[1], 27, 1); // Fundamental feedback v uporabi, nastaviti moram 8/9
prescaler (INT DELA tudi na 68)

                bitWrite (registers[1], 15, 1); // nastavi phase word na 1 (recommended)


//MUXOUT uporabljamo digital lock detect

                bitWrite (registers[2], 28, 1);

                bitWrite (registers[2], 27, 1);

                bitWrite (registers[2], 26, 0);


//nastavi charge pump tok 1.88mA pri Rset=5.1kOhm, glej tabelo v podatkovnem listu
```

```
//za loop filter BW glej simulacijo in oceni iz spektra, ce se ujema z 12,5kHz
                bitWrite (registers[2], 12, 0);
                bitWrite (registers[2], 11, 1);
                bitWrite (registers[2], 10, 0);
                bitWrite (registers[2],  9, 1);

//ker imamo 32MHz referenco moramo nastaviti hitrejše izbiranje VCO banda
                bitWrite (registers[3], 23, 1); //band select clock divider 1
//nastavimo delilnik na 128, tako da je hitrost ure za izbiro VCO 250kHz(max=500kHz)
                bitWrite (registers[4], 12, 0);
                bitWrite (registers[4], 13, 0);
                bitWrite (registers[4], 14, 0);
                bitWrite (registers[4], 15, 0);
                bitWrite (registers[4], 16, 0);
                bitWrite (registers[4], 17, 0);
                bitWrite (registers[4], 18, 0);
                bitWrite (registers[4], 19, 1);

//nastavi izhodno moc na +5dBm
                bitWrite (registers[4], 4, 1);
                bitWrite (registers[4], 3, 1);

//ugasni RF izhod DB5=0 (samo za test delovanja, nekaj signala mora vseeno priti ven)
//       bitWrite (registers[4], 5, 0);

//ugasni RF izhod dokler PLL ni zaklenjena (MTLD)
                bitWrite (registers[4], 10, 1);

//nastavitve specificne za INT-N ali FRAC-N nacin delovanja
                if (FRAC==0) {  // smo v INT-N mode (enke)
                        bitWrite (registers[2], 7, 1); //LDP
                        bitWrite (registers[2], 8, 1); //LDF INT-N
                        bitWrite (registers[3], 22, 1); //ABP na 3ns
                        bitWrite (registers[3], 21, 1); // Charge cancelation na 1 za INT-N
                }
                else {  //smo v FRAC-N mode (nicle)
```

```
                        bitWrite (registers[2], 7, 0); //LDP

                        bitWrite (registers[2], 8, 0); //LDF FRAC-N

                        bitWrite (registers[3], 22, 0); //ABP na 6ns

                        bitWrite (registers[3], 21, 0); // nastavi CP cancel za manjse spicke (samo za
INT-N)

                }


//tu lahko dodamo se kaksne vpise v registre po zelji, drugace ostanejo v veljavi zacetne vrednosti na
zacetku programa

//bitWrite (registers[x], y, z);

//kjer je x stevilka registra, y zaporedna stevilka bita in z vrednost bita


//izpis registrov, ki jih bomo poslali v cip preko serijske konzole (za kontrolo)

Serial.print("REG[0]: ");

Serial.print(registers[0], HEX);

Serial.print("\r\n");

Serial.print(registers[0], BIN);

Serial.print("\r\n");


Serial.print("REG[1]: ");

Serial.print(registers[1], HEX);

Serial.print("\r\n");

Serial.print(registers[1], BIN);

Serial.print("\r\n");


Serial.print("REG[2]: ");

Serial.print(registers[2], HEX);

Serial.print("\r\n");

Serial.print(registers[2], BIN);

Serial.print("\r\n");


Serial.print("REG[3]: ");

Serial.print(registers[3], HEX);

Serial.print("\r\n");

Serial.print(registers[3], BIN);

Serial.print("\r\n");
```

```
Serial.print("REG[4]: ");
Serial.print(registers[4], HEX);
Serial.print("\r\n");
Serial.print(registers[4], BIN);
Serial.print("\r\n");

Serial.print("REG[5]: ");
Serial.print(registers[5], HEX);
Serial.print("\r\n");
Serial.print(registers[5], BIN);
Serial.print("\r\n\n");

//izvedba vseh nastavitev in cakanje na ukaze iz tipk
        moc=f2m(RFout); //moc dobi vrednost glede na frekvencno obmocje
        SetADF4351(); // nastavi registre ADF4351
        RFints = RFint; //za detekcijo spremembe frekvence
        izpisLCD();
}
// izpis moci nosilca, ki mu pristevamo vrednost slabilca
        if (att != atts) {
                moc = (moc - (att - atts));
                atts = att; //za detekcijo spremebe slabljenja
                SetATT();
                izpisLCD();
        }

//funkcije za upravljanje s tipkami
//imamo stiri tipke in 5 moznosti izbiranja(levo, desno, gor, dol in gor+dol hkrati)
//s katerimi se pomikamo po LCD in spreminjamo, ter preverjamo mozne vrednosti spremenljivk
tipka = beri_tipke();
switch (tipka) {

        case DESNA:
                pcurs++;
                if (line == 0) {
                        if (pcurs == 9 ) {
```

```
                pcurs = 10;
                line = 0;
             }
            if (pcurs == 13 ) {
                pcurs = 3;
                line = 1;
            }
        }
        if (line == 1) {
        if (pcurs == 5) {
                pcurs = 9;
                line = 1;
            }
            if (pcurs == 10) {
                pcurs = 11;
                line = 1;
            }
            if (pcurs == 12) {
                pcurs = 5;
                line = 0;
            }
        }
        lcd.setCursor(pcurs, line);
        break;

case LEVA:
        pcurs--;
        if (line == 0) {
        if (pcurs == 9) {
                pcurs = 8;
                line = 0;
            }
        if (pcurs == 4) {
                pcurs = 11;
                line = 1;
            }
```

```
                }
                if (line==1) {
                        if (pcurs == 10) {
                                pcurs = 9;
                                line=1;
                        }
                        if (pcurs == 8) {
                                pcurs = 4;
                                line=1;
                        }
                        if (pcurs == 2) {
                                pcurs = 12;
                                line = 0;
                        }
                }
                lcd.setCursor(pcurs, line);
                break;


        case GOR:
        if (line == 0) {
                if (pcurs == 5) {
                        RFint = RFint + 1000000;
                        if ((RFint >= 2200000) && (RFint < 4400000)) { //skrajsaj ponavljanja v
funkcijo, ce bo cas...
                                RFint = RFint + 8;
                                za = RFint/8;   //zaokrozi velikost koraka, ki je na danem obmocju
mozen za uporabnikovo izbiro
                                                //(2-4-8kHz)
                                RFint = za * 8;
                        }
                        if ((RFint >= 1100000) && (RFint < 2200000)) {
                                RFint = RFint + 4;
                                za = RFint/4;
                                RFint = za * 4;
                        }
                        if ((RFint >= 550000) && (RFint < 1100000)) {
                                RFint = RFint + 2;
```

```
                za = RFint/2;
                RFint = za * 2;
        }
}
if (pcurs == 6) {
        RFint = RFint + 100000 ;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za = RFint/8;
                RFint = za*8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za = RFint/4;
                RFint = za*4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
                za = RFint/2;
                RFint = za * 2;
        }
}
if (pcurs == 7) {
        RFint = RFint + 10000 ;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za = RFint/8;
                RFint = za * 8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za = RFint/4;
                RFint = za * 4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
```

```
                za = RFint/2;
                RFint = za * 2;
        }
}
if (pcurs == 8) {
        RFint = RFint + 1000 ;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za = RFint/8;
                RFint = za * 8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za = RFint/4;
                RFint = za * 4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
                za = RFint/2;
                RFint = za * 2;
        }
}
if (pcurs == 10) {
        RFint = RFint + 100;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za = RFint/8;
                RFint = za * 8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za = RFint/4;
                RFint = za * 4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
```

```
                za = RFint/2;
                RFint = za * 2;
        }
}
if (pcurs == 11) {
        RFint = RFint + 10 ;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za = RFint/8;
                RFint = za * 8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za = RFint/4;
                RFint = za * 4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
                za = RFint/2;
                RFint = za * 2;
        }
}
if ((pcurs == 12) && (RFint >= 2200000) && (RFint < 4400000)) {
        RFint = RFint + 8 ;
        za = RFint/8;
        RFint = za * 8;
}
if ((pcurs == 12) && (RFint >= 1100000) && (RFint < 2200000)) {
        RFint = RFint + 4;
        za = RFint/4;
        RFint = za * 4;
}
if ((pcurs == 12) && (RFint >= 550000) && (RFint < 1100000)) {
        RFint = RFint + 2;
        za = RFint/2;
        RFint = za * 2;
```

```
        }
        if ((pcurs == 12) && (RFint < 550000) && (RFint >=34375)) RFint = RFint + 1;


        if (RFint > 4400000) RFint = RFints;
}
if (line == 1) {
        if (pcurs == 11) {
                eprom++;
        if (eprom == 10) eprom = 0;
        if (ZEE == 0) {
                        RFint = EEPROMReadlong(eprom * 4); // preberi frekvenco iz flasha
                        if (RFint > 4400000) RFint = 4400000;
                }
}
        if ((pcurs == 9) && (ZEE == 1)) {
                ZEE = 0;
        }
        else if ((pcurs == 9) && (ZEE == 0)) {
                ZEE = 1;
        }
        if (pcurs == 4) {
                att--;
        }
        if (pcurs == 3) {
                att = att - 10;
        }
        if (att > 63) att = 0;
}
izpisLCD();
break;


case DOL:
if (line == 0) {
        if ((pcurs == 12) && (RFint <= 550000) && (RFint > 34375)) RFint = RFint - 1;
        if ((pcurs == 12) && (RFint <= 1100000) && (RFint > 550000)) {
                RFint = RFint - 2;
```

```
        za=RFint/2;
        RFint=za*2;
}
if ((pcurs == 12) && (RFint <= 2200000) && (RFint > 1100000)) {
        RFint = RFint - 4;
        za=RFint/4;
        RFint=za*4;
}
if ((pcurs == 12) && (RFint <= 4400000) && (RFint > 2200000)) {
        RFint = RFint - 8;
        za=RFint/8;
        RFint=za*8;
}
if (pcurs == 11) {
        RFint = RFint - 10 ;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za=RFint/8;
                RFint=za*8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za=RFint/4;
                RFint=za*4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
                za=RFint/2;
                RFint=za*2;
        }
}
if (pcurs == 10) {
        RFint = RFint - 100 ;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za=RFint/8;
```

```
                RFint=za*8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za=RFint/4;
                RFint=za*4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
                za=RFint/2;
                RFint=za*2;
        }
}
if (pcurs == 8) {
        RFint = RFint - 1000;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za=RFint/8;
                RFint=za*8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za=RFint/4;
                RFint=za*4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
                za=RFint/2;
                RFint=za*2;
        }
}
if (pcurs == 7) {
        RFint = RFint - 10000;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za=RFint/8;
```

```
                RFint=za*8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za=RFint/4;
                RFint=za*4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
                za=RFint/2;
                RFint=za*2;
        }
}
if (pcurs == 6) {
        RFint = RFint - 100000;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za=RFint/8;
                RFint=za*8;
        }
        if ((RFint >= 1100000) && (RFint < 2200000)) {
                RFint = RFint + 4;
                za=RFint/4;
                RFint=za*4;
        }
        if ((RFint >= 550000) && (RFint < 1100000)) {
                RFint = RFint + 2;
                za=RFint/2;
                RFint=za*2;
        }
}
if (pcurs == 5) {
        RFint = RFint - 1000000;
        if ((RFint >= 2200000) && (RFint < 4400000)) {
                RFint = RFint + 8;
                za=RFint/8;
```

```
                        RFint=za*8;
                }
                if ((RFint >= 1100000) && (RFint < 2200000)) {
                        RFint = RFint + 4;
                        za=RFint/4;
                        RFint=za*4;
                }
                if ((RFint >= 550000) && (RFint < 1100000)) {
                        RFint = RFint + 2;
                        za=RFint/2;
                        RFint=za*2;
                }
        }
        if (RFint < 34375)  RFint = RFints;
        if (RFint > 4400000)  RFint = RFints;
}
if (line == 1) {
        if (pcurs == 11) {
                eprom--;
        if (eprom == 255) eprom = 9;
        if (ZEE == 0) {
                        RFint=EEPROMReadlong(eprom * 4); // preberi frek iz flasha
                        if (RFint > 4400000) RFint = 4400000;
                }
}
        if ((pcurs == 9) && (ZEE == 1)) {
                ZEE = 0;
        }
        else if ((pcurs == 9) && (ZEE==0)) {
                ZEE = 1;
        }
        if (pcurs == 4) {
                att++;
        }
        if (pcurs == 3) {
                att = att + 10;
```

```
                }
                if (att > 63) att = 63;
        }
          izpisLCD();
          break;

          case IZBERI:
          if (ZEE==1 && line == 1 && pcurs == 11) {
                  EEPROMWritelong(eprom * 4, RFint);
                  EEPROM.write(111 , 77); // za kontrolo, ce je bilo prvic vpisano v flash
                  lcd.setCursor(0 , 1);
                  lcd.print("  ZAPISANO  ");
          }
          lcd.setCursor(pcurs,line);
          delay(100);
          izpisLCD();
          break;

          default:
          break;
    }
}
```

Programska koda 2

```
//koda samo za testiranje registrov

#include <SPI.h>

#define LE 47

#define MUX 46

#define PDRF 48

#define CE 49



uint32_t registers[6] =  {0x449F58, 0x800BE81, 0x18008042, 0x8004B3, 0xA8043C, 0x580005} ; //550,006MHz
R=2, PFD=16MHz  <---- BOLJSE!!

//uint32_t registers[6] =  {0x898018, 0x8009F41, 0x18010042, 0x8004B3, 0xA8043C, 0x580005} ;
//550,006MHz R=4, PFD=8MHz  <---- BOLJSE, a vecji fazni sum!!

void setup()  {

  pinMode(MUX, INPUT);

  pinMode(LE, OUTPUT);

  pinMode(PDRF, OUTPUT);

  pinMode(CE, OUTPUT);

  digitalWrite(CE, 1);//chipe enable

  digitalWrite(PDRF, 1);//RFout enable

  digitalWrite(LE, HIGH);



  SPI.begin();                          // Init SPI bus

  SPI.setDataMode(SPI_MODE0);           // CPHA = 0 podatki na pozitivno stopnico ure

  SPI.setBitOrder(MSBFIRST);            // vrstni red

}

void WriteRegister32(const uint32_t vrednost)   //32bit registri

{

  digitalWrite(LE, LOW);

  delayMicroseconds(1);

  for (int i = 3; i >= 0; i--)          //  4 x 8bits=32

  SPI.transfer((vrednost >> 8 * i) & 0xFF); // bajt in maska prek SPI

  digitalWrite(LE, HIGH);

  delayMicroseconds(2);

  digitalWrite(LE, LOW);

}

void SetADF4351()

{ for (int i = 5; i >= 0; i--)  //registri R5-R0
```

```
    WriteRegister32(registers[i]);
}

void loop() {
  SetADF4351();
  delay(50000); //pavza mora bit, drugace bo prehitro nalagalo
}
```

Viri in literatura

[1] ADF4351 Wideband Synthesizer with Integrated VCO, podatkovni list, dostopno na http://www.analog.com/media/en/technical-documentation/data-sheets/ADF4351.pdf, (15. 5. 2016)

[2] HMC307 1dB LSB GaAs MMIC 5-BIT DIGITAL ATTENUATOR, podatkovni list, dostopno na http://www.analog.com/media/en/technical-documentation/data-sheets/hmc307.pdf, (15. 5. 2016)

[3] GALI5+ Surface Mount Monolithic Amplifier, podatkovni list, dostopno na https://www.minicircuits.com/pdfs/GALI-5+.pdf, (15. 5. 2016)

[4] HD44780 Dot Matrix Liquid Crystal Display Controller/Driver, podatkovni list, dostopno na https://www.sparkfun.com/datasheets/LCD/HD44780.pdf, (14. 4. 2016)

[5] Analog Devices, Fundamentals of Phase Locked Loops (PLLs), spletna publikacija, dostopno na http://www.analog.com/media/en/training-seminars/tutorials/MT-086.pdf, (15. 6. 2016)

[6] ADP150 Ultralow Noise, 150 mA CMOS Linear Regulator, podatkovni list, dostopno na http://www.analog.com/media/en/technical-documentation/data-sheets/ADP150.pdf, (15. 5. 2016)

[7] LM1117 800mA Low-Dropout Linear Regulator, podatkovni list, dostopano na http://www.ti.com/lit/ds/symlink/lm1117.pdf, (15. 5. 2016)

[8] Analog Devices, ADIsimPLL, PLL synthesizer design and simulation tool, dostopno na https://form.analog.com/form_pages/rfcomms/adisimpll.aspx, (23. 5. 2016)

[9] Amidon, feritni obročki tip FB-43-101, dostopno na http://www.rf-microwave.com/en/shop/0/170-ferrite-beads/2002-PF-02.html (15. 7. 2016)

[10] Analog Devices, Floating The HMC307QS16G For Positive Bias Operation, podatkovni list, dostopno na http://www.analog.com/media/en/technical-documentation/application-notes/floating_the_hmc307qs16g_for_positive_bias_operation.pdf, (6. 8. 2016)

[11] 4053 Triple 2-channel analog multiplexer/demultiplexer, podatkovni list, dostopno na http://www.nxp.com/documents/data_sheet/74HC_HCT4053.pdf, (13. 8. 2016)

[12] 74HC595 8-bit serial-in/serial or parallel-out shift register, podatkovni list, dostopno na https://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf, (17. 5. 2016)

[13] LM828 Switched Capacitor Voltage Converter, podatkovni list, dostopno na http://www.ti.com/lit/ds/symlink/lm828.pdf, (22. 8. 2016)

[14] Dan McMahill, Coplanar Waveguide Analysis/Synthesis Calculator, dostopno na http://wcalc.sourceforge.net/cgi-bin/coplanar.cgi, (15. 8. 2016)

[15] RC1587M Adjustable Low Dropout Linear Regulator, podatkovni list, dostopno na http://media.digikey.com/pdf/Data%20Sheets/Fairchild%20PDFs/RC1587_Rev2001.pdf, (5. 8. 2016)

[16] M. Vidmar, ATNC za Ne-Brezhibni Protokol, napajalnik za ATNC, spletna publikacija, dostopno na http://lea.hamradio.si/~s53mv/nbp/atnc/OpisATNC.pdf, (17.7. 2016)

[17] A. Fort, ADF4351 driven by an Arduino, spletna publikacija, dostopno na http://f6kbf.free.fr/html/ADF4351_LCD_07032016.zip, (8. 6. 2016)

[18] Analog Devices, ADF435x source code, dostopno na https://ez.analog.com/thread/13743#52311, (8. 6. 2016)

[19] Arduino, študijski pripomočki za Arduino razvojno okolje, dostopno na http://playground.arduino.cc/Main/ManualsAndCurriculum, (8. 6. 2016)

[20] M. Curtin, P. O'Brien, Phase-Locked Loops for High-Frequency Receivers and Transmitters–Part 1, Analog Dialogue, 1999, Volume 33, dostopno na http://www.analog.com/library/analogDialogue/cd/vol33n1.pdf, (28.6. 2016)

[21] M. Curtin, P. O'Brien, Phase-Locked Loops for High-Frequency Receivers and Transmitters–Part 2, Analog Dialogue, 1999, Volume 33, dostopno na http://www.analog.com/library/analogDialogue/cd/vol33n1.pdf, (28.6. 2016)

[22] M. Curtin, P. O'Brien, Phase-Locked Loops for High-Frequency Receivers and Transmitters–Part 3, Analog Dialogue, 1999, Volume 33, dostopno na http://www.analog.com/library/analogDialogue/cd/vol33n1.pdf, (28.6. 2016)